Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 2, No. 2, August 2016.

A HARDWARE-EFFICIENT BLOCK MATCHING UNIT FOR H.265/HEVC MOTION ESTIMATION ENGINE USING BIT-SHRINKING

Osama M. F. Abu-Sharkh[1] and Esam A. AlQaralleh[2]

Department of Computer Engineering, Princess Sumaya University for Technology, Amman, Jordan.
{osama[1], qaralleh[2]}@psut.edu.jo

(Received: 28-Dec.-2015, Revised: 22-Mar.-2016, Accepted: 10-Apr.-2016)

ABSTRACT

The main objective of this work is to enhance the processing performance of the recently introduced video codec H. 265/HEVC. Since most of the computations of H. 265/HEVC still occur in the motion estimation engine which is inherited from its predecessor H.264/AVC, we propose a bit-shrinking approach with a modified logic functionality to design an efficient and simplified block matching unit that replaces the already used Sum of Absolute Differences (SAD) unit. The hardware complexity of the proposed unit itself is reduced and the number of its generated output bits is reduced as well which in turn simplifies all the subsequent units of motion estimation. The hardware complexity, the consumed power and the processing delay of the motion estimation engine are therefore reduced significantly with only marginal deterioration in both the bit-rate and the peak-signal-to-noise-ratios (PSNR) of the tested High Definition (HD) and Ultra-High Definition (UHD) H.265/HEVC compressed videos. We simulate our design using HM16.6 and perform system logic synthesis using Synopsys's Design Compiler, targeting ASIC, for evaluation purposes.

KEYWORDS

H.265 / HEVC, Ultra-High Definition, Motion Estimation, Video Coding, Bit-Truncation.

1. INTRODUCTION

For almost a decade, H.264/AVC [1] has been the *de facto* standard for video coding. Its impressive performance opened wide doors for watching and exchanging videos almost everywhere. In spite of the fact that H.264 can handle High Definition (HD) videos, the sizes of these videos using H.264/AVC are still a concern, especially when using smart handheld devices with limited storage and power constraints. Recently, H.265/HEVC [2] has been introduced in the literature to provide better video coding performance than the legacy H.264/AVC. The former can provide up to twice the compression ratio of the latter, while maintaining the same video quality. Rearticulating, the former can provide much better video quality than the latter for the same video compression ratio. This huge increase in performance is due to the many enhanced techniques and methodologies that have been introduced in H.265/HEVC. Some of these enhancements have tackled the block matching criterion of motion estimation, the powerful engine of video coding. Although, block matching still relies on the use of the Sum of Absolute Differences (SAD) in H.265/HEVC which is inherited from H.264/AVC, the maximum block size has been enlarged in the former to 64x64 pixels instead of 16x16 pixels. Thus, the number of computations and consequently the number of the used SAD components and its constructing logic gates vastly increased in H.265/HEVC. Hence, the hardware complexity, the processing delay and the power consumption of the logic gates

increased vastly as well. Taking the processing delay as an example, Grois et al. conducted experiments in a recent work [3] to compare the performance of H.264/AVC, VP9 [4] and H.265/HEVC. They reported that the typical total encoding time of VP9 is around 130 times higher than the total encoding time of H.264/AVC and 7.35 times lower than the total encoding time of H.265/HEVC for the same PSNR value. This means that the total encoding time of H.265/HEVC is 955.5 times higher than that of H.264/AVC for the same PSNR value, which is actually a huge increase in the encoding time. The motion estimation engine by itself consumes around 60% to 80% of the total encoding time of H.264/AVC [5], while it consumes around 80% of the total encoding time of H.265/HEVC [6]. Thus, a pragmatic strategy to elevate the performance of H.265/HEVC relies on enhancing the performance of the core component of the motion estimation engine which is the block matching unit.

The main focus of this work is the block matching unit of the motion estimation engine. We propose in this paper a simplified and hardware-efficient block matching unit that replaces SAD. The framework of the proposed work is the motion estimation engine of the state-of-the-art video codec, H.265/HEVC. The main contributions and the differences from others in the literature are explained thoroughly in the following section.

The rest of the paper is organized as follows. We survey some of the related work in the literature and clarify our contribution in section 2. We then discuss the proposed designs of the block matching unit in section 3. We provide numerical analysis and evaluations with discussion in section 4 and finally conclude the paper in section 5.

2. LITERATURE REVIEW AND CONTRIBUTION

Many works in the literature have considered simplifying either the software and/or the hardware design of motion estimation in order to enhance the performance of video processing. Fast matching algorithms such as three steps search [7], four steps search [8], hexagon-based search [9], diamond search [10] and adaptive rood pattern search [11] tend to shrink the number of matched macro-blocks immensely. Most of these algorithms are only suitable for software implementations which are not as efficient as hardware implementations for real-time applications. Online arithmetic [12] and saturation arithmetic [13] are used to reduce the computational complexity of SAD. Nevertheless, the tree-adder implementation, where the SAD computations are performed, is still either time consuming or hardware costly. Vanne et al. performed in [14] arithmetic operations accompanied with several early termination mechanisms and sophisticated SAD computation control. Another approach based on SAD [15] takes the difference pixel count (DPC) as the selection criterion. Yeo et al. [16] use an XOR function instead of adders to simplify the matching criterion. One of the most recent promising approaches is to reduce the pixel resolution from eight bits to fewer bits. The works introduced in [17]-[20] use a one-bit transform by converting video frames into a single-bit-plane. On the other hand, the works introduced in [21]-[22] follow a bit-truncation approach where least significant bits are eliminated to simplify the hardware. One-bit transform and bit-truncation approaches deteriorate both the compression ratio and the video quality. The amount of deterioration in the latter depends on the number of truncated bits. Recently, Manjunatha and Sainarayanan recommended in [23] the use of a 1-bit full adder which consists of XOR, AND and OR gates instead of the commonly used 1-bit full adder which consists of XOR and NAND gates for performing SAD. In their work, they showed enhancements in the consumed power, latency and area when using the former rather than the latter. Following a different approach, we introduced in [24] a modified XOR function that replaces conventional SAD of the motion estimation engine of H.264/AVC. We showed enhancements over many proposed designs in the literature. All of the aforementioned authors who evaluated their works based on video coding, adopted the motion estimation engine of H.264/AVC with a maximum block size of 16x16 and low quality videos, CIF and QCIF in their evaluations.

Some recent works, which consider the enhancements of the motion estimation engine of H.265/HEVC, have also been proposed in the literature. We discuss some of the efforts which consider hardware implementations as follows. Sanchez et al. evaluated in [25] the use of the Multi-Point Diamond Search (MPDS) algorithm [26] in H.265/HEVC. They found that it is more hardware-friendly than the Enhanced Predictive Zonal Search (EPZS), which is implemented in the standard, on the expense of small amounts of deterioration in the video quality and compression ratio. Sinangil and Sze proposed in [27] a new hardware-aware search algorithm for HEVC motion estimation. They reported enhancements in area and bandwidth when using their algorithm. Jaja et al. proposed in [28] two fast motion estimation algorithms based on the structure of the triangle and the pentagon for H.265/HEVC. In their experimental evaluations, they found that the proposed algorithms can offer up to 63% and 61.9% speed-up in run-time when compared with the original algorithms of the standard. Medhat et al. proposed in [29] a Fast Center Search Algorithm (FCSA) for H.265/HEVC. They indicated that FCSA reaches average time saving ratio up to 40% for HD video sequences with insignificant loss in PSNR and compression ratio. Miyazawa et al. introduced in [30] a complete hardware implementation for H.265/HEVC and evaluated its efficiency in processing videos in real-time applications. They compared their implementation with a professional-use H.264/AVC video encoder available in the market. They were able to encode HD videos at 60fps in real-time. Pastuszak and Trochimiuk proposed in [31] a high-throughput motion estimation system to process Ultra-High Definition (UHD) videos in H.265/HEVC. The system embeds two parallel processing paths for the integer-pel and the fractional-pel motion estimation. Their synthesis showed that the system is able to encode UHD videos at 30fps with only small deteriorations in PSNR and compression ratio. Ye et al. proposed in [32] a parallel clustering tree search (PCTS) algorithm for integer-pel motion estimation that processes the prediction units (PU) simultaneously with a parallel scheme. The hardware implementation of PCTS can support quad-full HD (QFHD) videos at 30fps in real-time. All of the aforementioned efforts still rely on the use of the block matching unit, conventional SAD in designing the algorithms to enhance the motion estimation of H.265/HEVC.

In this paper, we introduce several enhancements on our previous work [24]. They are mentioned in the following. Our new modified block matching unit is implemented in the motion estimation engine of H.265/HEVC which has a different hardware-efficiency with the increased block size than the one used by H.264/AVC. We also introduce in this paper a new bit-shrinking approach to reduce the number of generated output bits of the matching unit which is reflected on all subsequent stages of the motion estimation engine. We perform system logic synthesis using Synopsys's Design Compiler [33], targeting ASIC, to evaluate our design and compare it with the conventional SAD and other works in the literature. We consider the number of gates, the consumed power and the processing delay as performance metrics for evaluation purposes. The obtained results show the superiority of our design in all performance metrics. We also run extensive simulations using HM16.6 [34] to measure the video quality and the compression ratio. We apply our simulations on both HD and UHD videos for evaluation purposes and consider a block size of 64x64.

It is worthy to mention that the proposed block matching unit can be utilized by many motion estimation algorithms in the literature, such as the ones proposed in [25]-[32], to replace conventional SAD and hence enhance performance.

3. PROPOSED BLOCK MATCHING UNIT

Let us first give a brief description of the sum of absolute differences operation. SAD is a measure of similarity between a block of pixels of the present frame and a block of pixels of a previous reference frame of a video. It estimates motion between the considered frames to remove redundant information and consequently reduces the sizes of the videos. Considering NxN as the macro-block size, $p_k(i,j)$ as the current pixel of a macro-block, $p_{k-1}(i+u, j+v)$ as

the candidate pixel of a macro-block of a reference frame and $[-q, q-1]$ as the search range, SAD is defined by the following equation.

$$SAD(u,v) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} |p_k(i,j) - p_{k-1}(i+u, j+v)|; \quad (1)$$

where $-q \le u, v \le q-1$.

The absolute difference function in Equation (1) can be described as:

$$|X-Y| = \begin{cases} X + \bar{Y} + 1 & if\ MSB = 0 \\ \bar{X} + Y + 1 & if\ MSB = 1 \\ 0 & if\ X = Y \end{cases} \quad (2)$$

The sum of absolute difference function involves the comparison of two pixels. These pixels have unsigned representations of luminous values. The unsigned nature of these pixels shall burden the system. To avoid this, the sum of absolute difference function is designed in several ways in hardware. One way is by subtracting two unsigned numbers and then making a decision about the obtained result by converting the negative sum into positive as shown in Figure 1 with the aid of an XOR gate. Another design is achieved by subtracting the first number from the second one and also the second number from the first one, simultaneously, and then selecting the positive result via a multiplexer. The former implementation encounters longer delay than the latter, since the critical path goes through two adders in the former while it goes only through one adder and one multiplexer in the latter. Note that the two adders in the second design operate in parallel.

There are several hardware implementations for the full adder. The basic, simplest and mostly used implementation is the Ripple Carry Adder (RCA). As the name indicates, the carry ripples from the n^{th} bit to the next one and so on until it reaches the most significant bit. Thus, the most significant bit of the output cannot be calculated until all the preceding bits are calculated one after another. It can be deduced that the more the number of calculated bits is, the more is the delay in generating the final result. Another implementation of a full adder is the Carry Look Ahead Adder (CLA). The latter is faster than the RCA, but more complex, which leads to a larger implementation area and more power consumption.

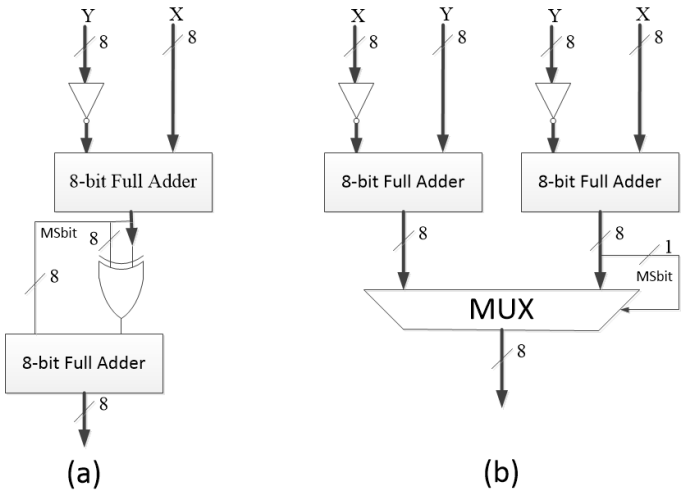

Figure 1. Two different designs for the sum of absolute difference circuit.

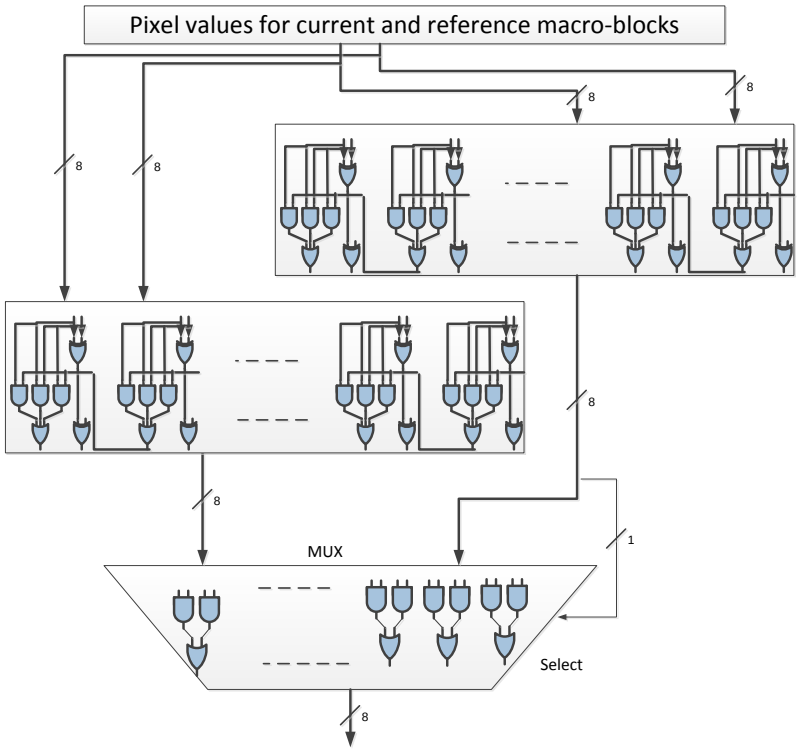

Figure 2. Sum of absolute difference circuit using 8-bit full adders and a multiplexer.

Considering the circuit shown in Figure 1. (b), the sum of absolute difference function for one pixel matching is implemented using two 8-bit full adders and one multiplexer. These units are built from many logic gates as shown in Figure 2. Each macro-block matching consists of many SAD operations. Since very large amounts of macro-blocks' matching occur for motion estimation during the processing of a video, gigantic amounts of SAD operations are performed. For example, it requires almost 66,846,720 SAD operations or 534,773,760 SAD operations to process only one single frame of an HD video with the resolution of 1920x1080 or an UHD video with the resolution of 3840x2160, respectively, using a 64x64 macro-block size and a [−64, 63] search range. Furthermore, performing SAD operations on different block sizes shall increase these numbers by multiples. This is hardware-costly and also makes the video processing encounter considerable delay and power consumption. Hardware implementation with parallel SAD operations is the ultimate solution to meet the real-time constraints when using conventional SAD in full search motion estimation.

Bit-truncation has been proposed in [21]-[22] as a promising approach to reduce the cost of the conventional SAD unit. The main concept of bit-truncation is to perform conventional SAD

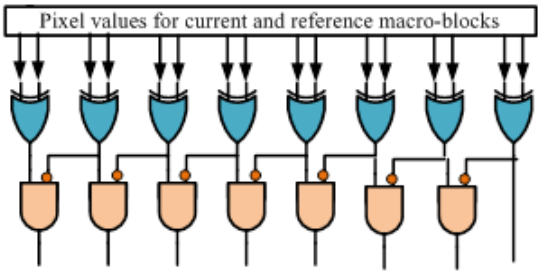

Figure 3. Matching unit without bit-shrinking (MXOR).

The number of generated output bits is 8.

operations on pixels using fewer number of bits by truncating the least significant bits. It is described by:

$$BT_m(u,v) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} |p_k\,(i,j)_{7:m} - \; p_{k-1}\,(i+u, j+v)_{7:m}|; \tag{3}$$

where $p_k\,(i,j)_{7:m}$ is the current pixel of a macro-block with mth least significant bits truncated, $p_{k-1}\,(i+u, j+v)_{7:m}$ is the candidate pixel of a macro-block of a reference frame with mth least significant bits truncated. The macro-block size is $N \times N$, the search range is $[-q, q-1]$ and $-q \le u, v \le q-1$.

The more the number of truncated bits is, the simpler is the SAD operation. As can be deduced from Equation (3), the bit-truncation approach does not really simplify the full adder circuit itself which makes the propagation delay only shortened by the number of truncated bits. Bit-truncation approach also requires a special memory design to support such variable number of bits which are fed into the SAD unit [21].

Considering the hardware-costly SAD and its long processing time and large power consumption, we introduced in [24] a block matching unit with modified XOR functionality that produces almost the same outputs of the SAD unit for the many different combinations of the inputs and is built from a much smaller number of logic gates. It is described by:

$$MXORU(u,v) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \widehat{XOR}(p_k\,(i,j), p_{k-1}\,(i+u, j+v)); \tag{4}$$

where,

$$\widehat{XOR} = \sum_{m=0}^{7} (2^m)\left(XOR\left(b_m^{p_k}, b_m^{p_{k-1}}\right)\right).\overline{\left(XOR\left(b_{m-1}^{p_k}, b_{m-1}^{p_{k-1}}\right)\right)}; \tag{5}$$

where $b_m^{p_k}$ is the m^{th} bit of a pixel in the k^{th} frame. We denote the matching unit, described by Equation (5) and shown in Figure 3, by MXOR.

The proposed unit outperforms SAD in terms of hardware-complexity, processing delay and power consumption as explained briefly below.

In a regular Ripple Carry Adder, which is used in the computation of SAD as mentioned before, the highest significant bit of a result depends on the carry which is generated after summing all lower significant bits. This encounters a long delay, since the carry keeps propagating through the adder as explained earlier. In the proposed matching unit, the computation of each output bit depends only on the neighbouring lower significant bit. Thus, the propagation of bits is limited to only one bit position. This saves a considerable amount of time in producing the result of the matching process. Furthermore, the circuit of the proposed matching unit which evaluates each bit is composed of only an XOR gate, an inverter and an AND gate. Hence, the number of constructing logic gates of the proposed unit is enormously less than that of SAD. This is clearly illustrated when observing and comparing the units shown in Figure 2 and Figure 3. The simplification in designing the unit leads to a shorter processing delay along with a smaller unit area and less power consumption.

Since most - and not all - of the output bits of SAD and the modified XOR match, a deterioration in the performance of the video codec shall occur. By implementing the modified

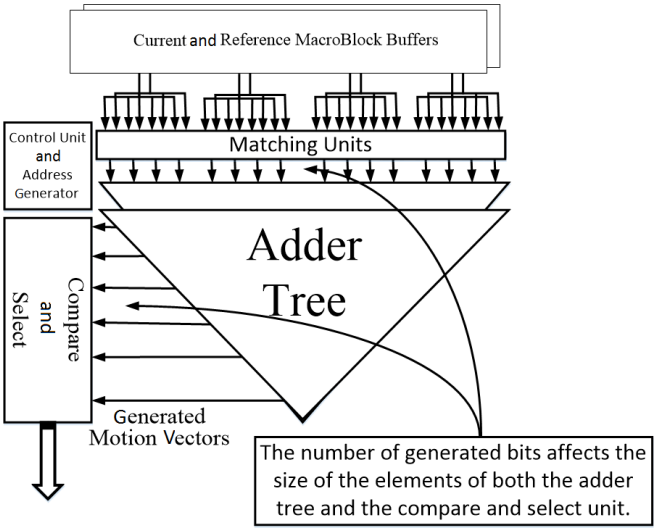

Figure 4. A typical motion estimation engine with a matching unit, an adder tree and a compare and select unit.

XOR in the motion estimation engine of H.264/AVC video codec and taking sample CIF and QCIF videos, we showed in [24] that the amounts of deterioration in both video quality and compression ratio are marginal. In this work, we introduce a new approach, bit-shrinking, to further reduce the computational burden of motion estimation based on the previously introduced MXORU. It is explained as follows.

Bit-shrinking reduces the number of the generated output bits of the matching unit. This is directly reflected on the following stages; namely the parallel adder tree and the compare and select unit shown in Figure 4. The purpose of the adder tree is to accumulate the generated output bits of the matching unit for each sub-block. Therefore, reducing the number of the output bits will obviously reduce the complexity of the processing elements and the used registers in the following stages. Hence, the hardware architecture is simplified and all the related performance metrics in terms of processing delay, power consumption and hardware cost are reduced.

Considering Equation (4) as the general form representation of the block matching unit, the $\widehat{XOR}$ represents several circuit designs according to the level of bit-shrinking. To simplify the Boolean representation of the $\widehat{XOR}$ function of the matching unit, let us first define Ω_m and Ψ_m as:

$$\Omega_m = XOR\left(b_m^{p_k}, b_m^{p_{k-1}}\right), \tag{6}$$

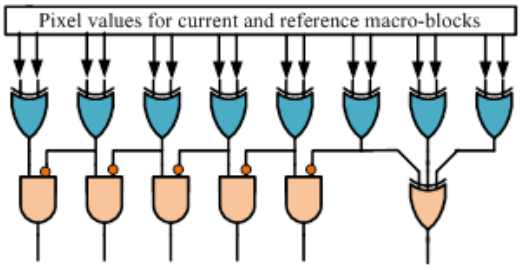

Figure 5. Matching unit with two-bit-shrinking (MXOR2). The number of generated output bits is reduced to 6.

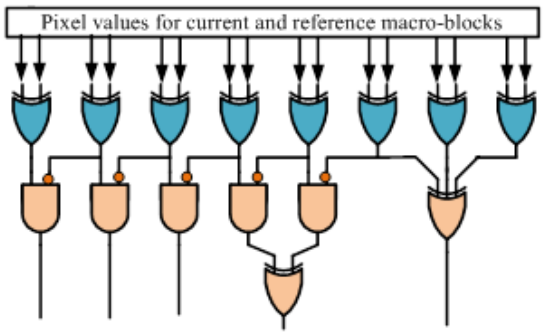

Figure 6. Matching unit with three-bit-shrinking (MXOR3). The number of generated output bits is reduced to 5.

and

$$\Psi_m = \left(XOR\left(b_m^{p_k}, b_m^{p_{k-1}}\right)\right).\overline{\left(XOR\left(b_{m-1}^{p_k}, b_{m-1}^{p_{k-1}}\right)\right)}; \tag{7}$$

where $b_m^{p_k}$ is the m^{th} bit of a pixel in the k^{th} frame.

First, by combining the lowest three output bits of the unit shown in Figure 3 through an XOR gate, one bit is generated. It represents the least significant bit of the output and is described by $2^0 XOR(\Omega_0, \Omega_1, \Omega_2)$. The remaining bits are shifted by 2^{m-2}. Thus, the number of the generated output bits is reduced from eight to six. The matching unit is denoted by MXOR2 and shown in Figure 5. It is also described by:

$$\widehat{XOR} = 2^0 XOR(\Omega_0, \Omega_1, \Omega_2) + \sum_{m=3}^{7} (2^{m-2})\,\Psi_m. \tag{8}$$

Combining the 2nd and 3rd output bits of the unit shown in Figure 5 through another XOR gate reduces the number of the generated output bits to five rather than six.

The least significant bit is described by $2^0 XOR(\Omega_0, \Omega_1, \Omega_2)$. The next higher significant bit, which is generated based on the new combination, is described by $2^1 XOR(\Psi_3, \Psi_4)$ and the remaining output bits are shifted by 2^{m-3}. The matching unit is denoted by MXOR3 and shown in Figure 6. It is also described by:

$$\widehat{XOR} = 2^0 XOR(\Omega_0, \Omega_1, \Omega_2) + 2^1 XOR(\Psi_3, \Psi_4) + \sum_{m=5}^{7} (2^{m-3})\,\Psi_m. \tag{9}$$

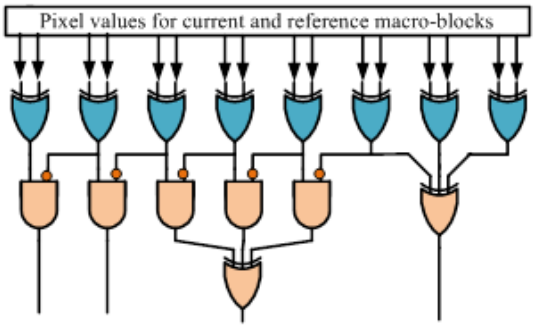

Figure 7. Matching unit with four-bit-shrinking (MXOR4). The number of generated output bits is reduced to 4.

Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 2, No. 2, August 2016.

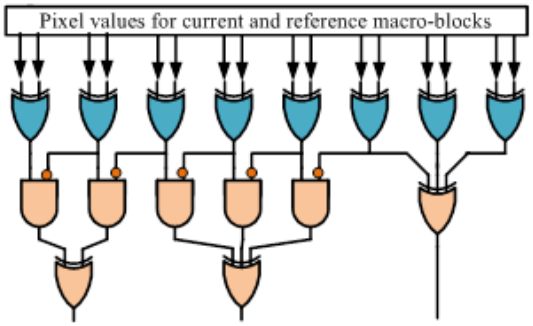

Figure 8. Matching unit with five-bit-shrinking (MXOR5). The number of generated output bits is reduced to 3.

In a different manipulation, combining the 2nd, 3rd and 4th output bits of the unit shown in Figure 5 through an XOR gate reduces the number of the generated output bits to four.

The least significant bit is described by $2^0XOR(\Omega_0, \Omega_1, \Omega_2)$. The next higher significant bit, which is generated based on the new combination of the three bits, is described by $2^1XOR(\Psi_3, \Psi_4, \Psi_5)$ and the remaining output bits are shifted by 2^{m-4}. The matching unit is denoted by MXOR4 and shown in Figure 7. It is also described by:

$$\widehat{XOR} = 2^0XOR(\Omega_0, \Omega_1, \Omega_2) + 2^1XOR(\Psi_3, \Psi_4, \Psi_5) + \sum_{m=6}^{7} (2^{m-4})\,\Psi_m. \qquad (10)$$

Furthermore, combining the two most significant output bits of the unit shown in Figure 7 through an XOR gate reduces the number of the generated output bits to only three bits.

The least significant bit is described by $2^0XOR(\Omega_0, \Omega_1, \Omega_2)$. The next higher significant bit is described by $2^1XOR(\Psi_3, \Psi_4, \Psi_5)$ and finally the most significant bit, which is generated based on the new combination, is described by $2^2XOR(\Psi_6, \Psi_7)$. The matching unit is denoted by MXOR5 and shown in Figure 8. It is also described by:

$$\widehat{XOR} = 2^0XOR(\Omega_0, \Omega_1, \Omega_2) + 2^1XOR(\Psi_3, \Psi_4, \Psi_5) + 2^2XOR(\Psi_6, \Psi_7). \qquad (11)$$

The generated output bits of the proposed matching units deviate from the original generated output bits of SAD. The more the shrinking is, the more is the deviation. This leads to a degradation in the coding performance of the video codec which is illustrated by deteriorations in both the bit-rate and PSNR. In the following section, we show that the amounts of deterioration are marginal and that our proposed approach actually pays off.

4. NUMERICAL ANALYSIS AND EVALUATIONS

First, we perform ASIC system logic synthesis using Synopsys's Design Compiler [ver I-2013.12-SP5-10 using TSMC 90nm general-purpose nominal-threshold-voltage library] for the proposed matching units, the conventional SAD, the modified implementation of SAD [23] (we denote it by MISAD) and finally the matching units of the bit-truncation approach. We consider the latter, since this approach is the closest to our proposed bit-shrinking approach. Taking the number of generated output bits as a reference, our units described by Equations (8), (9), (10) and (11) are analogous to the units which generate six output bits (NTB2), five output bits (NTB3), four output bits (NTB4) and three output bits (NTB5), respectively, of the bit-truncation approach.

We consider the following performance metrics for evaluation purposes: the hardware complexity in terms of the number of two-input NAND gates, the processing delay in terms of the critical path in nano-second and the consumed power in micro-watt. The power estimation is under a global operating voltage of 1.1V, a load capacitance of 0.2 pF, an operating frequency of 100MHz and with a medium confidence level. Table 1 shows the obtained results. From the table, there is a huge increase in performance when using all the proposed matching units when compared with the conventional SAD. The number of two-input NAND gates, consumed power and processing delay of MXOR, which generate the same number of output bits of the conventional SAD, are only 18%, 19% and 14% of the number of two-input NAND gates, consumed power and processing delay, respectively, of the conventional SAD, while their values increase to 24%, 24% and 31% of the conventional SAD, respectively, when considering MXOR5 which generates only 3 output bits. Table 1 also shows that the results of MISAD are very close to the results of the conventional SAD. The number of two-input NAND gates, consumed power and processing delay of MISAD are 96.5%, 98.7% and 90.9% of the number of two-input NAND gates, consumed power and processing delay, respectively, of the conventional SAD, which indicates a very modest improvement of this contemporary work [23]. Table 1 also shows the superiority of our matching units when compared with the matching units of the bit-truncation approach with the same number of generated output bits. Thus, in terms of hardware efficiency illustrated by the adopted three metrics, our proposed matching units outperform the conventional SAD, MISAD and all the matching units of the bit truncation-approach.

The benefits gained by bit-shrinking are not limited to the matching unit only, as we mentioned before, but it also affects later stages of motion estimation engine (shown in Figure 4). Bit-shrinking shall reduce the complexity of the adder tree in terms of a lower-width adder and smaller accumulating registers for each sub-block in 64x64 macro-block. Furthermore, the compare and select unit shall also be reduced in terms of the sizes of the compare units as well as the used registers. For example, the size of each accumulating register reduces from 12 bits, 14 bits, 16 bits, 18 bits and 20 bits for 8-bit pixel matching in a conventional SAD for sub-block sizes of 4x4, 8x8, 16x16, 32x32 and 64x64, respectively, down to only 7 bits, 9 bits, 11 bits, 13 bits and 15 bits, respectively, when using MXOR5.

The significant performance elevation introduced by our units may boost the widespread of H.265/HEVC, especially in modest devices such as smartphones which have storage and power constraints. Nevertheless, we have to show that the compression ratio and video quality are not much affected by the proposed units to support our statement. Therefore, we perform intensive We perform the simulations using HEVC reference software HM16.6 to compare the video quality illustrated by PSNR and the compression ratio illustrated by bit-rate. The PSNR is the

Table 1. Comparison of all matching units in terms of consumed power, number of two-input NAND gates and processing delay.

	SAD	***MXOR***		***MXOR2***		***MXOR3***		***MXOR4***		***MXOR5***	
Number of gates	174	32	(18%)	35	(20%)	37	(21%)	43	(25%)	42	(24%)
Critical path (ns)	20.23	2.87	(14%)	4.2	(21%)	4.42	(22%)	6.18	(31%)	6.18	(31%)
Power (μW)	13.1858	2.5583	(19%)	2.7648	(21%)	2.8554	(22%)	3.0737	(23%)	3.227	(24%)
	SAD	***MISAD***		***NTB2***		***NTB3***		***NTB4***		***NTB5***	
Number of gates	174	168	96.5%	108	(62%)	100	(57%)	86	(49%)	54	(31%)
Critical path (ns)	20.23	18.39	90.9%	18.12	(90%)	17.35	(86%)	16.66	(82%)	13.29	(66%)
Power (μW)	13.1858	13.015	98.7%	10.3675	(79%)	7.2736	(55%)	5.367	(41%)	4.203	(32%)

simulations taking into consideration both the compression ratio and video quality for different kinds of frame sequences of the standard.

most frequently used indicator by the research community to measure picture quality. It is defined by:

$$PSNR = 20 \log \frac{255}{\sqrt{MSE}}; \quad (12)$$

where 255 is the largest pixel value for 8-bit representation and MSE is the Mean Square Error between a noise-free M×N monochrome frame I and its noisy approximation K. It is defined by:

$$MSE = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} [I\,(i,j) - K(i,j)]^2. \quad (13)$$

The simulations are conducted first on six HD videos: Tennis, Beauty, Bosphorus, Honey Bee, Jockey and Ready Steady Go. The resolution of these videos is 1080p. We consider main profile level 6.2 with Random-Access (IBBB frame sequences) and Low-Delay-P (IPPP frame sequences) configurations. The block search range is [−64, 63] with maximum CU size 64 and maximum CU partition depth 4 with full search motion estimation. The number of frames taken in each sequence is 120. The simulations are carried on Windows 8os platform with Intel i7 extreme @ 2.93GHz CPU and 8GB RAM.

Table 2 shows the obtained results of PSNR and the bit-rate of a conventional SAD as well as the deviations in these two metrics when using all matching units compared with SAD. As expected and discussed in the previous section, due to the partial mismatch of the generated output bits between the proposed units and the conventional SAD, a deterioration occurs in both PSNR and bit-rate. Still, the amount of decrease in PSNR and the amount of increase in bit-rate are very small as frankly illustrated by the results obtained from the simulations performed on all the tested videos. Among our proposed matching units, the maximum deviation of PSNR occurs when processing Tennis video with a value of -0.04dB using MXOR5, while the maximum increase in bit-rate also occurs when processing Tennis video with a value of 1.5%. These values are considered very small and hence the deteriorations are actually marginal even at the peaks.

Table 3 summarizes the deviations in PSNR and bit-rate by showing averages. Note that the amount of degradation in the performance is very small when using our approach compared with the conventional SAD. The amount of degradation in PSNR ranges from 0.001dB and 0.002dB when considering MXOR to 0.013dB and 0.018dB when considering MXOR5 for IPPP and IBBB frame sequences, respectively. On the other hand, the amount of increase in the bit-rate ranges from 0.09% and 0.07% when considering MXOR to 0.543% and 0.684% when considering MXOR5 for IPPP and IBBB frame sequences, respectively. It can be deduced from the table that the amounts of deterioration in both PSNR and bit-rate are small when using either bit-truncation or bit-shrinking approaches. Thus, we can confidently state that the video quality and the compression ratio are not practically affected when following these approaches. Note that MISAD performs the SAD operation with a different implementation of the building 1-bit full adder unit as explained earlier in Section 2. Therefore, the generated output bits of MISAD and the conventional SAD are exactly the same. Hence, there are no deteriorations in both bit-rate and PSNR when using MISAD instead of the conventional SAD.

To evaluate the proposed matching units in processing very high resolution videos, we also conduct simulations on 4K UHD videos with the resolution of 3840x2160. Note that UHD videos can be processed by H.265/HEVC and the open-source video codec VP9, but not H.264/AVC. The videos under test are Marathon, Library, Scarf and Traffic & Building. Table 4 summarizes the results by showing averages. Note that the amounts of deviation in both bit-

Table 2. The amounts of deterioration in PSNR (dB) and bit-rate of all units when compared with SAD. The results shown are for both sequences IPPP and IBBB (HD videos).

		SAD		***MXOR***		***MXOR2***		***MXOR3***		***MXOR4***	
VIDEO	***SEQ.***	*BR*	*PSNR*	*Δ-BR%*	*Δ-PSNR*	*Δ-BR%*	*Δ-PSNR*	*Δ-BR%*	*Δ-PSNR*	*Δ-BR%*	*Δ-PSNR*
TENNIS	***IPPP***	153342	37.1282	0.05%	-0.003	-0.38%	-0.018	0.33%	-0.009	0.75%	-0.017
	IBBB	931558	36.5691	0.30%	-0.008	0.06%	-0.030	0.64%	-0.021	1.22%	-0.038
BEAUTY	***IPPP***	439896	38.3652	0.08%	-0.003	-0.06%	-0.002	0.41%	-0.004	0.51%	-0.004
	IBBB	313972	38.0187	0.10%	-0.001	-0.08%	-0.002	0.29%	-0.005	0.49%	-0.007
BOSPHORUS	***IPPP***	397017	38.2873	0.10%	-0.002	0.12%	-0.001	0.00%	-0.013	0.03%	-0.021
	IBBB	316291	38.2566	-0.22%	-0.001	-0.27%	-0.003	-0.17%	-0.012	0.14%	-0.015
HONEYBEE	***IPPP***	98027	39.1785	0.23%	-0.012	0.49%	-0.005	-0.04%	-0.013	0.40%	-0.010
	IBBB	70723	39.7868	-0.07%	0.007	0.16%	-0.006	0.28%	-0.001	0.22%	-0.008
JOCKEY	***IPPP***	389278	39.5202	0.11%	0.014	-0.02%	0.013	0.53%	0.005	0.85%	0.002
	IBBB	305175	39.5179	0.27%	0.000	0.15%	-0.004	0.65%	-0.020	1.08%	-0.019
R.S.G.	***IPPP***	389278	37.3463	-0.01%	-0.001	0.09%	-0.010	0.27%	-0.011	0.77%	-0.013
	IBBB	305175	37.3082	0.06%	-0.008	0.15%	-0.019	0.20%	-0.015	0.89%	-0.011

		MXOR5		***NTB2***		***NTB3***		***NTB4***		***NTB5***	
VIDEO	***SEQ.***	*Δ-BR%*	*Δ-PSNR*	*Δ-BR%*	*Δ-PSNR*	*Δ-BR%*	*Δ-PSNR*	*Δ-BR%*	*Δ-PSNR*	*Δ-BR%*	*Δ-PSNR*
TENNIS	***IPPP***	0.82%	-0.02	-0.03%	0.000	-0.05%	-0.005	0.03%	-0.009	0.36%	-0.018
	IBBB	1.50%	-0.04	0.13%	-0.002	0.14%	-0.006	0.28%	-0.023	1.19%	-0.047
BEAUTY	***IPPP***	0.41%	0.00	0.02%	0.000	-0.01%	0.001	0.25%	-0.005	0.86%	-0.008
	IBBB	0.34%	-0.01	-0.11%	0.000	-0.04%	-0.002	0.27%	-0.007	0.89%	-0.014
BOSPHORUS	***IPPP***	-0.01%	-0.01	0.37%	0.000	0.19%	0.002	0.24%	-0.013	0.11%	-0.020
	IBBB	0.29%	-0.01	-0.20%	0.001	-0.17%	-0.005	-0.07%	-0.002	-0.11%	-0.014
HONEYBEE	***IPPP***	0.32%	-0.02	-0.06%	-0.005	-0.02%	-0.010	0.12%	-0.011	-0.02%	-0.006
	IBBB	-0.06%	0.00	0.20%	0.000	0.52%	0.001	0.46%	-0.001	0.36%	0.008
JOCKEY	***IPPP***	0.90%	-0.01	0.39%	0.008	0.21%	0.014	0.22%	-0.009	0.67%	0.002
	IBBB	1.01%	-0.01	0.09%	-0.004	0.22%	-0.003	0.40%	-0.010	0.77%	-0.019
R.S.G.	***IPPP***	0.83%	-0.02	0.05%	-0.004	0.05%	-0.010	0.02%	-0.011	0.32%	-0.010
	IBBB	1.01%	-0.03	-0.04%	0.003	-0.01%	-0.002	0.09%	-0.004	0.32%	-0.018

Table 3. The average amounts of deterioration in PSNR (dB) and bit-rate of the proposed units and the units of bit-truncation approach when compared with SAD (HD videos).

	MXOR		***MXOR2***		***MXOR3***		***MXOR4***		***MXOR5***	
SEQ.	*Δ-BR%*	*Δ-PSNR*	*Δ-BR%*	*Δ-PSNR*	*Δ-BR%*	*Δ-PSNR*	*Δ-BR%*	*Δ-PSNR*	*Δ-BR%*	*Δ-PSNR*
IPPP	0.09%	-0.001	0.04%	-0.004	0.25%	-0.007	0.55%	-0.010	0.543%	-0.013
IBBB	0.07%	-0.002	0.03%	-0.011	0.31%	-0.012	0.67%	-0.017	0.684%	-0.018

		NTB2		***NTB3***		***NTB4***		***NTB5***	
SEQ.		*Δ-BR%*	*Δ-PSNR*	*Δ-BR%*	*Δ-PSNR*	*Δ-BR%*	*Δ-PSNR*	*Δ-BR%*	*Δ-PSNR*
IPPP		0.122%	-0.0003	0.063%	-0.0014	0.147%	-0.0095	0.384%	-0.010
IBBB		0.012%	-0.0005	0.109%	-0.0029	0.237%	-0.0078	0.568%	-0.018

rate and PSNR are still very small when using the proposed matching units compared with the conventional SAD. Therefore, we can conclude that the proposed matching units can be utilized in the motion estimation engine of H.265/HEVC to process different resolution videos with marginal effect on the video quality and the compression ratio.

The obtained results of the extensive simulations that are performed on both HD and UHD videos using HM16.6 validate the correctness of the functionalities of the proposed matching units in replacing SAD. The videos have been processed and encoded successfully using H.265/HEVC with marginal deteriorations in the values of both PSNR and bit rate as illustrated in Table 3 and Table 4 and explained earlier. As a validation step for the Verilog code by which we describe the matching units and performed synthesis using Synopsys's Design Compiler as discussed earlier, we adopt a framework which was proposed in [35] that utilizes MATLAB/SIMULINK [36] and ModelSim [37] concurrently as a verification environment for the hardware design of the matching unit of the motion estimation engine. The verification environment co-simulates the hardware design under verification (DUV) along with its software model. It continuously reports mismatches, if existed, for each processed operation at the pixel level along with their occurrence times. In this work, we adjust the verification environment and implement the Verilog codes of the proposed matching units, the conventional SAD, MISAD and the matching units of the bit-truncation approach, one at a time, along with their codes used in HM16.6. The tests are performed on HD and UHD videos which are uploaded to MATLAB and fed to the verification environment. No mismatches are reported. This ensures the validity of the conducted work.

Table 4. The average amounts of deterioration in PSNR (dB) and bit-rate of the proposed units and the units of bit-truncation approach when compared with SAD (UHD videos).

	MXOR		***MXOR2***		***MXOR3***		***MXOR4***		***MXOR5***	
SEQ.	*Δ-BR%*	*Δ-PSNR*	*Δ-BR%*	*Δ-PSNR*	*Δ-BR%*	*Δ-PSNR*	*Δ-BR%*	*Δ-PSNR*	*Δ-BR%*	*Δ-PSNR*
IPPP	0.005%	-0.00165	-0.002%	-0.00182	0.021%	-0.0025	0.251%	-0.0039	0.236%	-0.005
IBBB	0.011%	-0.00053	0.087%	0.000425	0.040%	-0.00318	0.100%	-0.00795	0.155%	-0.008
			NTB2		***NTB3***		***NTB4***		***NTB5***	
SEQ.			*Δ-BR%*	*Δ-PSNR*	*Δ-BR%*	*Δ-PSNR*	*Δ-BR%*	*Δ-PSNR*	*Δ-BR%*	*Δ-PSNR*
IPPP			-0.07%	-0.00145	0.01%	-0.00068	0.06%	-0.00153	0.03%	-0.004
IBBB			-0.03%	-0.00253	0.04%	0.00015	0.02%	-0.0001	0.08%	-0.004

```
uiSum += abs( piOrg[0] - piCur[0] );
```

(a)

```
ABS  = piOrg[0] ^ piCur[0];
ABS1 = ~(ABS << 1) | 1;
MXOR = ABS & ABS1;
uiSum += MXOR;
```

(b)

Figure 9. (a) The code of conventional SAD operation of HM16.6 (b) The code of MXOR implemented in HM16.6.

5. CONCLUSION

We proposed in this paper a hardware-efficient block matching unit for H.265/HEVC motion estimation engine. We followed a bit-shrinking approach with a modified logic functionality to reduce the number of the building two-input NAND gates, consumed power and processing delay of the matching units. We performed extensive simulations to evaluate our design. We considered HD and UHD videos in our evaluations, since their usage have been spreading tremendously in recent years. The results obtained show the superiority of our approach over SAD, MISAD and the bit-truncation approach taking the three adopted performance metrics into consideration. The results show only small amounts of deterioration in video quality and compression ratio when shrinking the number of output bits. The amounts of increase in the number of two-input NAND gates, consumed power and processing delay when using MXOR units with bit-shrinking instead of the pure MXOR unit are small. Nevertheless, the MXOR units with bit-shrinking actually outperform the pure MXOR, since they significantly reduce the hardware complexity of all subsequent stages including the parallel adder tree and the compare and select units of the motion estimation engine.

It is important to clarify that we propose the matching units only for hardware implementations and not for software implementations. The design considers manipulating the pixels at the bit-level which makes the proposed units run efficiently on hardware. As mentioned above, considerable enhancements have been shown when we evaluated the hardware design of the proposed matching units and compared them with the hardware designs of others. Software implementations of the proposed matching units are not efficient, since extra computations are performed by the motion estimation engine when compared with the conventional SAD. The code shown in Figure 9 (a) is the code extracted from HM16.6 that performs one pixel matching using SAD, while the code shown in Figure 9 (b) is the code implemented in HM16.6 to perform the same operation using MXOR which is taken here as an example. The codes to compute MXOR2 through MXOR5 are even longer than the code of MXOR. Hence, extra time is needed by the software to perform the coding process when using our proposed units instead of the conventional SAD. We emphasize the fact that we only performed the software simulations using HM16.6 to validate the functionalities of the matching units and also to measure the amounts of deterioration in both PSNR and bit-rate when using the proposed units instead of SAD.

REFERENCES

[1] Advanced Video Coding, Rec. ITU-T H.264 and ISO/IEC 14496-10, 2014.

[2] High Efficiency Video Coding, Rec. ITU-T H.265 and ISO/IEC 23008-2, 2015.

[3] D. Grois et al., "Performance Comparison of H. 265/MPEG-HEVC, VP9 and H. 264/MPEG-AVC Encoders," Proc. Picture Coding Symp., pp. 394-397, 2013.

[4] VP9, WebM Project's Next-generation Open Video Codec, 2013.

[5] Y. Li, J. Xiao and W. Wu, "Motion Estimation Based on H.264 Video Coding," Proc. 5th Int. Congress Image and Signal Process., pp. 104-108, 2012.

[6] M.U.K. Khan, M. Shafique and J. Henkel, "AMBER: Adaptive Energy Management for On-chip Hybrid Video Memories," Proc. IEEE/ACM Int. Conf. Computer-Aided Design, pp. 405-412, 2013.

[7] R. Li, B. Zeng and M.L. Liou, "A New Three-step Search Algorithm for Block Motion Estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 4, no. 4, pp. 438-442, August 1994.

[8] L. M. Po and W. C. Ma, "A Novel Four Step Search Algorithm for Fast Block Motion Estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 6, no. 3, pp. 313–317, June 1996.

[9] C. Zhu et al., "A Novel Hexagon-based Search Algorithm for Fast Block Motion Estimation," Proc. Int. Conf. Acoustics, Speech and Signal Process., pp. 1593–1596, 2001.

[10] S. Zhu and K.K. Ma, "A New Diamond Search Algorithm for Fast Block-matching Motion Estimation," IEEE Trans. Image Process., vol. 9, no. 2, pp. 287–290, Feburuary 2000.

[11] Y. Nie and K. K. Ma, "Adaptive Rood Pattern Search for Fast Block-matching Motion Estimation," IEEE Trans. Image Process., vol. 11, no. 12, pp. 1442–1448, December 2002.

[12] J. Olivaresa et al., "SAD Computation Based on Online Arithmetic for Motion Estimation," Elsevier Microprocessors and Microsystems, vol. 30, no. 5, pp. 250–258, August 2006.

[13] T. H. Tran, H. M. Cho and S. B. Cho, "Performance Enhancement of Sum of Absolute Difference (SAD) Computation in H.264/AVC Using Saturation Arithmetic," Proc. Emerging Intelligent Computing Technol. and Applicat., pp. 396–404, 2009.

[14] J. Vanne et al., "A High-performance Sum of Absolute Difference Implementation for Motion Estimation," IEEE Trans. Circuits Syst. Video Technol., vol. 16, no. 7, pp. 876-883, July 2006.

[15] S. Lee, J. M. Kim and S. I. Chae, "New Motion Estimation Algorithm Using Adaptively Quantized Low Bit Resolution Image and Its VLSI Architecture for MPEG2 Video Encoding," IEEE Trans. Circuits Syst. Video Technol., vol. 8, no. 6, pp. 734–744, October 1998.

[16] H. Yeo and Y. H. Hu, "A Novel Architecture and Processor-level Design Based on a New Matching Criterion for Video Compression," Proc. IEEE Workshop on VLSI Signal Process. IX, pp. 448–457, 1996.

[17] S. Ertürk, "Multiplication-free One-bit Transform for Low-complexity Block-based Motion Estimation," IEEE Signal Process. Lett., vol. 14, no. 2, pp. 109-112, February 2007.

[18] A. Akin, Y. Dogan and I. Hamzaoglu, "High Performance Hardware Architectures for One Bit Transform Based Motion Estimation," IEEE Trans. Consum. Electron., vol. 55, no. 2, pp. 941–949, August 2009.

[19] A. Akin, G. Sayilar and I. Hamzaoglu, "High Performance Hardware Architectures for One Bit Transform Based Single and Multiple Reference Frame Motion Estimation," IEEE Trans. Consum. Electron., vol. 56, no. 2, pp. 1144–1152, July 2010.

[20] S. Chatterjee and I. Chakrabarti, "Low Power VLSI Architectures for One Bit Transformation Based Fast Motion Estimation," IEEE Trans. Consum. Electron., vol. 56, no.4, pp. 2652–2660, January 2011.

[21] A. Bahari, T. Arslan and A.T. Erdogan, "Low-power H. 264 Video Compression Architectures for Mobile Communication," IEEE Trans. Circuits Syst. Video Technol., vol. 19, no. 9, pp. 1251-1261, September 2009.

[22] I. Chakrabarti, K. Batta and S. Chatterjee, "Efficient Pixel Truncation Algorithm and Architecture," Motion Estimation for Video Coding, Studies in Computational Intelligence, Springer, vol. 590, pp. 65-83, 2015.

[23] D. V. Manjunatha and G. Sainarayanan, "Low-Power Sum of Absolute Difference Architecture for Video Coding," Emerging Research in Electronics, Computer Science and Technol., Lecture Notes in Electrical Engineering, Springer, vol. 248, pp. 335-341, 2014.

[24] E. AlQaralleh and O. M. F. Abu-Sharkh, "Low-complexity Motion Estimation Design Using Modified XOR Function," Springer Multimedia Tools and Applicat., DOI: 10.1007/s11042-015-2948-z, September 2015.

[25] G. Sanchez, M. Porto and L. Agostini, "A Hardware Friendly Motion Estimation Algorithm for the Emergent HEVC Standard and Its Low Power Hardware Design," Proc. 20th IEEE Int. Conf. Image Process., pp. 1991-1994, 2013.

[26] M. E. Sinangil et al., "Hardware-aware Motion Estimation Search Algorithm Development for High-efficiency Video Coding (HEVC) Standard," Proc. 19th IEEE Int. Conf. Image Process., pp. 1529-1532, 2012.

[27] G. Sanchez et al., "DMPDS: A Fast Motion Estimation Algorithm Targeting High Resolution Videos and Its FPGA Implementation," Int. J. Reconfigurable Computing, vol. 2012, pp. 1-12, January 2012.

[28] E. Jaja et al., "Efficient Motion Estimation Algorithms for HEVC/H. 265 Video Coding," Information Science and Applicat., Lecture Notes in Electrical Engineering, Springer, vol. 339, pp. 287-294, 2015.

[29] A. Medhat et al., "Fast Center Search Algorithm with Hardware Implementation for Motion Estimation in HEVC Encoder," Proc. 21st IEEE Int. Conf. Electronics, Circuits and Systems, pp. 155-158, 2014.

[30] K. Miyazawa et al., "Real-time Hardware Implementation of HEVC Video Encoder for 1080p HD Video," Proc. Picture Coding Symp., pp. 225-228, 2013.

[31] G. Pastuszak and M. Trochimiuk, "Algorithm and Architecture Design of the Motion Estimation for the H. 265/HEVC 4K-UHD Encoder," Springer J. Real-Time Image Process., DOI: 10.1007/s11554-015-0516-4, July 2015.

[32] X. Ye, D. Ding and L. Yu, "A Hardware-oriented IME Algorithm and Its Implementation for HEVC," Proc. IEEE Visual Commun. and Image Process. Conf., pp. 205-208, 2014.

[33] Synopsys, (2016, Mar 11), Design Compiler [Online], Available: https://www.synopsys.com/Tools/Implementation/RTLSynthesis/Pages/default.aspx.

[34] JCT-VC High Efficiency Video Coding Reference Software, (2016, Mar 11), HM 16.6 [Online], Available: https:/hevc.hhi.fraunhofer.de.

[35] E. A. AlQaralleh, O. M. F. Abu-Sharkh and B. A. Y. AlQaralleh, "MATLAB/Simulink-based Verification Environment for Motion Estimation in H. 264/AVC," Proc. 5th IEEE Int. Conf. Digital Inform. and Commun. Technol. and Its Applicat., pp. 59-63, 2015.

[36] Mathworks, (2016, Mar 11), Matlab/Simulink Users Guide, Application Program Interface Guide [Online], Available: http://www.mathworks.com.

[37] Mentor Graphics, (2016, Mar 11), ModelSim [Online], Available: https://www.mentor.com/products/fpga/model/

ملخص البحث:

يتمثل الهدف الأساسي من هذا البحث في تحسين المعالجة لِتقنية ترميز الصور عالية الفعالية التي تم إدخالها حديثاً ويرمز لها بـِ (H.265/HEVC). ونظراً لأنّ معظم الحسابات لتلك التقنية ما زالت تُجرى في محرّك تقدير الحركة المورث من التقنية السابقة المعروفة بـِ (H.264/AVC)، فقد اقترحنا نهجاً يقوم على تقليص عدد خانات الرقم الثنائي بوظيفيّة منطقية معدَّلة من أجل تصميم وحدةٍ لمواءمة الزُّمَر، فعّالة ومبسّطة، لتحلّ محلّ وحدة مجموع الفروق المطلقة المستخدمة سابقاً. وقد تم التقليل من تعقيد الأجزاء المادية للوحدة المقترحة، وكذلك التقليل من عدد ما تقوم بتوليده من خانات الرقم الثنائي الخَرج، الأمر الذي من شأنه أن يبسّط جميع الوحدات اللاحقة لتقدير الحركة.