GU J Sci 30(1): 69-78 (2017)

Gazi University

Journal of Science

http://dergipark.gov.tr/gujs

Efficient Hardware Architecture for Selective Gray Coded Bit Plane Based Low Complexity Motion Estimation

Muhammad ASLAM[1], Anıl ÇELEBİ[1,*]

[1] *Kocaeli University Laboratory of Integrated Systems (KUTSAL), University of Kocaeli, Dept. of Electronics and Telecom. Eng., Umuttepe Campus, 41380, Kocaeli,*

Article Info

Received: 23/08/2016
Accepted: 24/02/2017

Keywords

Motion estimation
Gray-coding
One-bit transform
Low-complexity ME

Abstract

In video compression, motion estimation is exploited to remove temporal redundancy. Computationally, motion estimation is one of the most expensive parts of a video encoder. In this work, efficient and novel hardware architecture is proposed to implement selective Gray-coded bit-plane based motion estimation algorithm. Spiral search algorithm is employed as search scheme in the novel hardware architecture. Experimental results show that considerable amount of hardware resources are saved thanks to the proposed architecture compared to the recent works in the literature.

1. INTRODUCTION

Today, electronics devices such as cell phones, cameras have the facility to capture videos. Efficient compression methods are needed for reducing the size of the data to be stored or transmitted via a communication channel. Motion Estimation (ME) is one of the key components of any video coding method to remove temporal redundancy between consecutive image frames in a video sequence. Although it has a significant impact on reduction of the size of the video data, ME occupies more than 75% of the total computational power of a typical video encoding infrastructure [1]. In Block-based ME, Current frame is split into equally sized blocks and none of them overlap each other. Hardware architecture for full search ME scheme must produce and calculate motion vectors (MV) for all respective macroblocks within a current frame. General equation for deciding motion vectors is shown in (1).

$$SSD(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{M-1}\{I^c(x,y) - I^r(x+u,y+v)\}^2 \qquad (1)$$

$$-s \le u,v \le s$$

$$MV = \min(SSD(u,v))$$

Where $I^c(x,y)$ and $I^r(x+u,y+v)$ show current and reference block pixels, respectively. Additionally, (u,v) and s represents candidate MV and search range respectively. The potential MV that provides the minimum Sum of Squared Error (SSD) represents the MV of the related block. Previously proposed ME methods available in the literature mostly use Sum of Absolute Differences (SAD) as matching criterion instead of SSD [2, 3] as shown in (2).

$$SAD(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{M-1}|I^c(x,y) - I^r(x+u,y+v)| \qquad (2)$$

$$-s \le u,v \le s$$

$$MV = \min(SAD(u,v))$$

*Corresponding author, e-mail: anilcelebi@kocaeli.edu.tr

Processing power needed for SAD process is considerably high thus, several approaches are proposed in the literature to reduce this load. In [4] a Bit Plane Matching (BPM) technique is proposed as matching criterion in ME operation. In BPM based methods, Boolean exclusive-OR (XOR) is used as matching criteria in place of SAD to considerably reduce the power consumption and hardware resources. In BPM, generally, the Number of Non-Matching Points (NNMP) criterion is utilized for matching as shown in (3).

$$NNMP(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{M-1}\{B^c(x,y) \oplus B^r(x+u,y+v)\}$$
$$-s \le u,v \le s \tag{3}$$
$$MV = \min(NNMP(u,v))$$

Where $B^c(x,y)$ and $B^c(x\ +u,y+v)$ represent the pixels in the current bit plane and reference bit plane respectively in this equation. Logical XOR operation is represent by $\oplus$. NNMP values decide the MV of the respective Current Block (CB). The candidate location providing the minimum NNMP value is considered as MV of the CB. Although the operation shown in (3) looks quite simple, binarization process requires additional hardware which is usually not mentioned in the related works.

Several low complexity BPM based ME methods are introduced previously such as single Bit Plane (BP) based ME as in [4], 2-Bits Transform (2BT) based ME or constrained 1-bit transform based ME where 2 BPs are exploited for ME process in [5] and [6] respectively. As in [7, 8, 9], different hardware architectures are proposed for single BP or 2BT based ME approaches, but none of them provide any solution for the binarization.

Truncated Gray Coded Bit Plane Matching (TGCBPM) method is introduced to considerably simplify the binarization process [10]. In TGCBPM based ME method, the bit planes that are used in ME are generated by truncating the unused least significant Gray coded bit planes. Equation shown in (4) is utilized for calculating NNMP values in TGCBPM method.

$$NNMP_{TGC}(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{M-1}\sum_{n=NTB}^{N-1} 2^{N-NTB}\{\, g_n^t(x,y) \oplus g_n^{t-1}(x+u,y+v)\,\} \tag{4}$$
$$-s \le u,v \le s$$
$$MV = \min\left(NNMP_{TGC}(u,v)\right)$$

Where N, NTB represent the number of bits in a pixel and the number of truncated bits respectively. Experiments reveal that best results are achieved when setting NTB=5, which means only 3 Most Significant Bit (MSB) planes are utilized in ME [10]. In general, low bit-depth based ME techniques helps to achieve more efficient hardware architecture compared to full bit depth based ME methods [7-12].

Recently, 4 MSB planes are used to obtain one-bit plane by using a bit selection scheme presented in [13] to further reduce the number of utilized bit planes in ME process. In [14], a new binarization scheme is presented to obtain binary image by using 3 MSB planes efficiently and this method is called Selective Gray-Coded Bit-Plane based Binarization Approach (SGC-BPBA). Experiments show that SGC-BPBA is more efficient compared to the binarization method proposed in [13]. In this paper, novel and more efficient hardware architecture is presented for the method in [14] to further reduce the hardware complexity and power consumption. Furthermore, it is now possible to implement the ME algorithm proposed in [14] for bigger block size such as 64×64 in high efficiency video coding (HEVC) thanks to the proposed hardware architecture.

2. PROPOSED HARDWARE ARCHITECTURE

In this paper, an efficient hardware architecture is presented for the ME algorithm that is introduced in [14]. The proposed hardware architecture is mainly composed of five blocks: memory unit, multiplexer array, 2-dimensional Processing Elements (PE) array, parallel counter and comparator as shown in Figure 1.

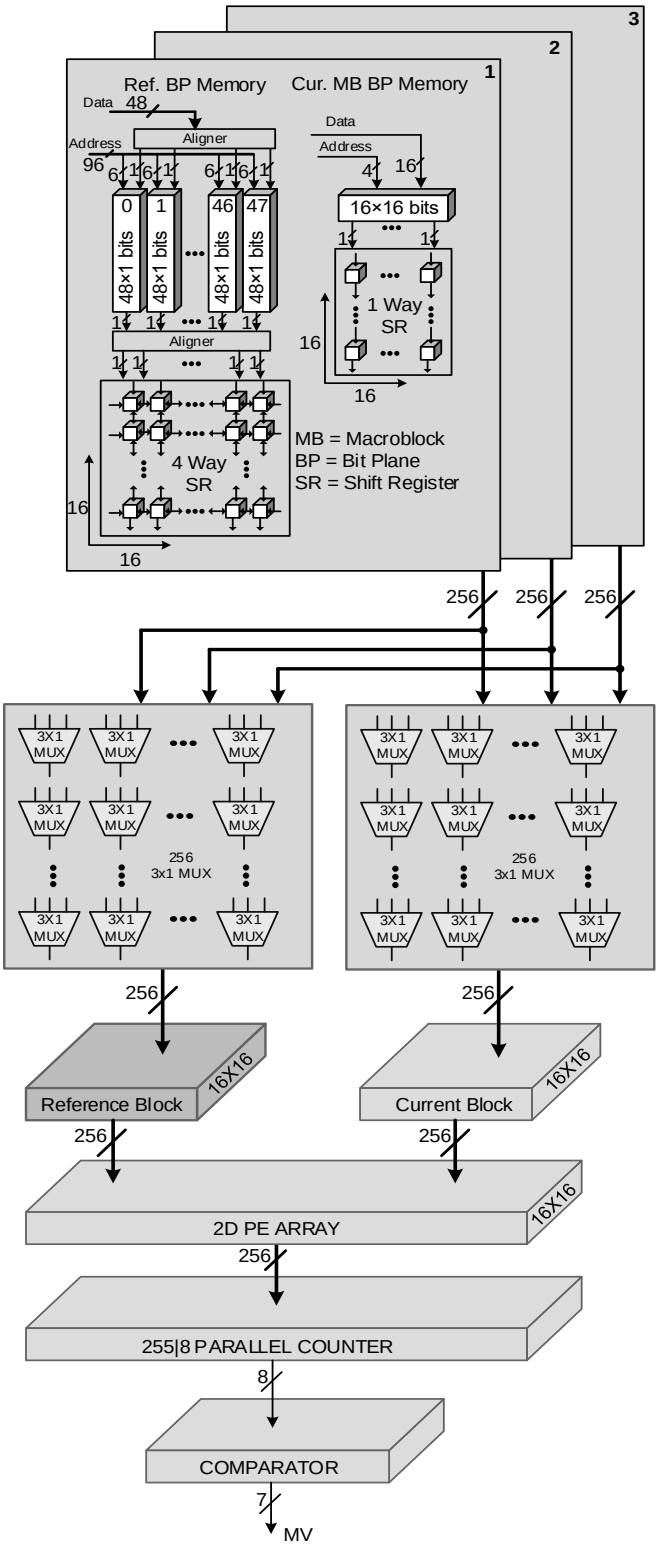

Figure 1. *Proposed hardware architecture*

2.1. Memory Unit

Memory unit consists of four sub-blocks; reference window memory, current macroblock memory, two data aligner blocks, and two Shift Register (SR) arrays. In the proposed hardware architecture, one 16×16 sized RAM and forty-eight 48×1 RAMs are utilized for current block memory and reference window respectively. As there are 3 MSB planes used in the proposed scheme [14], three memory modules are utilized in the proposed hardware architecture.

2.1.1. Fast search algorithm support

In proposed architecture, reference window data is passed through respective aligner before storing the pixel data into the relevant memory block. The aligner allows the incoming data to be stored in ladder shaped manner as proposed in [2]. This scheme allows reading any row or column from a single memory in one clock. This type of memory architecture allows an arbitrary move in vertical as well as horizontal directions in the search window. In other words, it supports fast ME algorithms.

Figure 2 shows a sample data pattern for memory architecture used in proposed hardware architecture. It shows memory architecture for [-4,4) search range. Darkly shaded area represents 0^{th} column and lightly shaded area represents 0^{th} row. It can be seen that both row and column can be read at one clock cycle thanks to this organization in the utilized memory block. After reading from RAM, the data is passed through another aligner that re-shifts the data appropriately. For example, if 1^{st} row is read then the incoming data will be shifted one pixel to the left by the aligner.

Address	MEM0	MEM1	MEM2	MEM3
0	0 0	0 1	0 2	0 3
1	1 3	1 0	1 1	1 2
2	2 2	2 3	2 0	2 1
3	3 1	3 2	3 3	3 0
4	4 0	4 1	4 2	4 3
5	5 3	5 0	5 1	5 2
6	6 2	6 3	6 0	6 1
7	7 1	7 2	7 3	7 0
8	0 4	0 5	0 6	0 7
9	1 7	1 4	1 5	1 6
10	2 6	2 7	2 4	2 5
11	3 5	3 6	3 7	3 4
12	4 4	4 5	4 6	4 7
13	5 7	5 4	5 5	5 6
14	6 6	6 7	6 4	6 5
15	7 5	7 6	7 7	7 4

Figure 2. *Sample pattern for search range* $[-4,4)$

16×16 shift register is used to store the current block pixels. Since, 3 MSB planes are utilized in the proposed method in [14], 3 bits SR array is utilized to store the current block pixels. At the beginning of ME operation, during loading process, the current block pixels are shifted into the SR array. No more shift operations are performed on current block SR array after loading process is completed.

The SR array used to store Reference Block (RB) is capable of moving the data up, right, left and down directions. In the proposed architecture, fast search algorithm and more than one level of data reuse schemes are supported thanks to spiral search scheme and ladder shaped memory architecture respectively. Data reuse's effects on hardware architecture are explained in detail in [15]. Various data reuse schemes can be utilized to lessen the bandwidth of off chip memory. For example, performing ME for CBs concurrently is an indirect way to realize the data reuse but this scheme needs higher number of PEs [8]. Another way to realize data reuse is using fast search algorithms as in [2]. The data reuse scheme that is exploited in the presented hardware architecture is shown in Figure 3. As illustrated in this figure, both horizontal and vertical movements are performed in the search window. In each vertical move, one new pixel row is read from memory and 15 old rows are reused by exploiting the SR array. In the same way during horizontal move, one new pixel column is read from the memory and previous 15 columns are reused.

Spiral like movement is used in the proposed architecture as shown in Figure 4 [14]. 4 way SR array makes this kind of movement possible as shown in Figure 1. Main advantage of spiral movement is that it allows performing early termination or adaptive search range techniques in future.

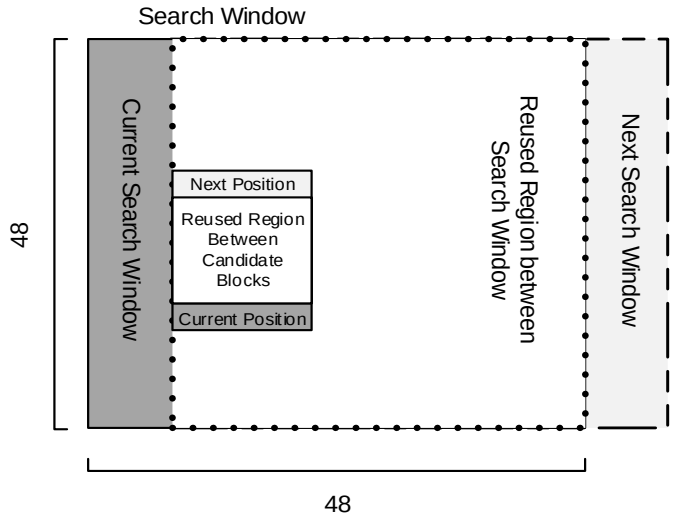

Figure 3. *Data reuse scheme*

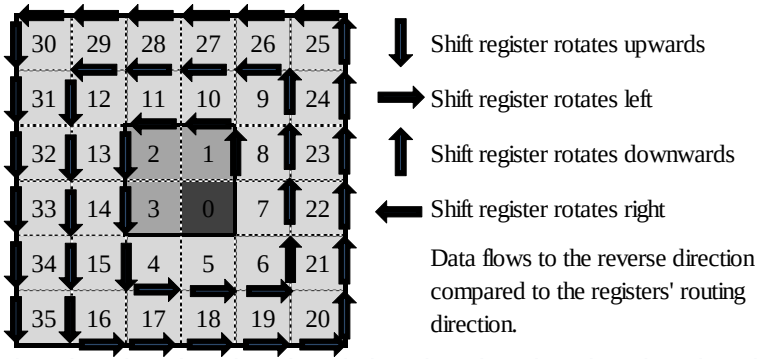

Figure 4. *Spiral search diagram*

2.2. Multiplexer Array

In [14], selective Gray coded bit plane based binarization approach is presented to obtain single bit plane by using a selective approach on the three Gray coded bit planes. 3×1 MUX array is exploited to implement that scheme in the proposed hardware architecture as shown in Figure 1. Since the block size is 16×16 pixels, 256 3×1 multiplexers are used in the proposed architecture. Detailed diagram of a MUX in the multiplexer array is shown in Figure 5, in which mode selector is used to select the appropriate bit from Gray coded three bits driven into this block according to the method proposed in [14].

2.3. Pe Array

16×16 pixels size RB, after appropriate bit selection performed by the multiplexer block, is driven to 2D PE array as shown in Figure 1. In 2D PE array, RB and CB are compared by using Boolean XOR operation.

2.4. Parallel Counter And Comparator

The Parallel Counter (PC) is used to compute the NNMP between RB and CB. The architecture of PC consists of seven stages of sub PCs of size 3|2, 7|3, 15|4, 31|5, 63|6, 128|7 and, 255|8 respectively as shown in Figure 6. PC has 255 inputs while a macroblock has 256 pixels. Experiments show that absence of one pixel does not have a considerable impact on the ME performance [12]. Additionally, architecture of PC also becomes simpler comparatively. The output of counter is followed by a comparator to find out the MV of the CB by comparing the NNMP values being generated sequentially by this block.

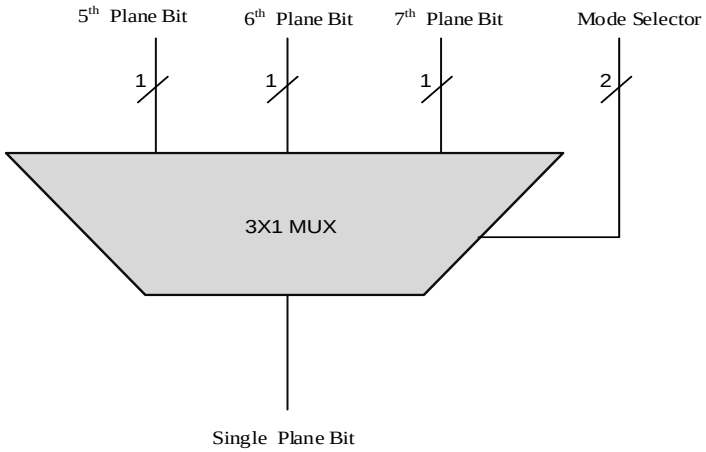

Figure 5. *Bit Plane Selector*

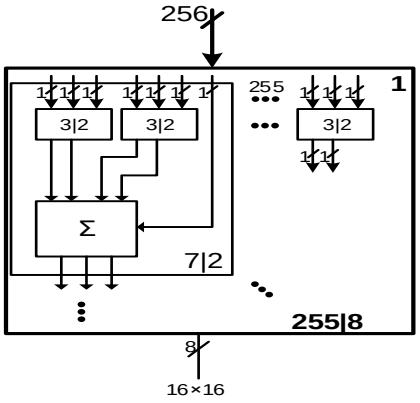

Figure 6. *Parallel Counter Architecture*

3. IMPLEMENTATION RESULTS

28 nm (Xilinx Zynq 7000 - Package Code: xc7a200tfbg676-2) FPGA device is used to implement the proposed hardware architecture. This is the FPGA used in [14] thus, a fare comparison is performed. Synthesis results show that the proposed architecture occupies 7587 LUTs, 2219 DFFs and 24 BRAMs, which is the 5.64%, 0.83% and 6.58% of the total available resources of the target FPGA device, respectively. The maximum frequency at which the proposed architecture can operate is 150MHZ for the selected FPGA device. A comprehensive comparison of the proposed hardware architecture and the architectures proposed in [7, 14, 16, 17] is shown in Table 1. Thanks to the proposed architecture, utilization ratio of LUTs and DFFs is decreased by 6.62% and 69.82% as compared to [14] as shown in Table 1 respectively. Proposed architecture is superior compared to the architecture in [14] in terms of all parameters except maximum clock frequency according to the numbers given in Table 1.

Power analysis of the proposed architecture is performed after post-implementation functional simulation for the same block numbers used in [14] to achieve a fare comparison. The logic and signal columns refer to the power consumption of the logic resources and routing resources of the target FPGA respectively. For example, a configurable logic block in an FPGA is a logic resource on the other hand a switch matrix is a

routing resource. The power estimation tool can group power data according to the characteristic of the specific hardware resource such as logic, signal, clock etc.

D type flip flops based SR array and BRAM are utilized in [14] and the proposed hardware architecture in this paper, respectively, to store current block and reference block (search window) pixels. Power consumption comparison between proposed architecture and [14] is shown in Table 2. It is important to note that average power consumption values for the proposed hardware architecture and [14] are 8+7.78 = 15.78mW and 9.11+7.89 = 17mW, respectively for 20ns clock period. It shows that average power consumption of the proposed architecture is decreased by 7.18% as compared to [14]. Similarly, the decrement of the average power consumption is 35.44% for 10ns clock period. The main reason for the power consumption reduction is the elimination of shift register array in which D type Flip Flops (DFF) are utilized. It is a known fact that a 1 bit BRAM memory cell consumes less power compared to more than 3 times bigger DFF based 1-bit memory cell. 28 nm (Xilinx Zynq 7000 - Package Code: xc7z100ffv1156-2) is also utilized for power analysis of the proposed hardware architecture for observing the effects of different FPGAs on its power consumption and results are shown in Table 3. Table 2 and Table 3 show that there is no significant power consumption difference. From these results, it can be concluded that hardware complexity of the proposed hardware architecture does not change considerably between different FPGAs.

The control of a DFF based memory implementation is easier compared to the BRAM based memory organization. Implementation of spiral search is easier with DFF based SR array but this architecture cannot be synthesized even for the state of the art FPGA chips because of the huge wire congestion caused by routing between flip flops. Because of that BRAM resources are utilized since they are hardcoded into the FPGA chip. The bottleneck caused by BRAMs are their control. Data should be well scheduled and organized for implementation of spiral search scheme. BRAM resources of the target FPGA is used in the proposed architecture to store the data in a ladder shaped manner thanks to which a row or column can be read in one clock cycle. This feature is achieved by DFFs based shift registers in [14]. The synthesis process for the architecture which includes shift registers based on DFFs is higher comparatively because of the high wire congestion. It is observed that synthesis time for the architecture proposed in [14] takes roughly 3 times more compared to the proposed architecture. Since the number of wires increase exponentially with an increase on the block size, synthesis time ratio between two architectures is also expected to increase exponentially. As a result, proposed architecture seems more promising in terms of the CPU time needed for synthesis, placement and routing phases. Thus, the proposed architecture can be integrated within an HEVC encoder easier compared to [14]. The total CPU time for synthesis of the proposed architecture is 95 seconds whereas this number is 325 for the architecture proposed in [14] for a block size of 16×16. The PC used for the synthesis is a MAC Book Pro with Intel i7 CPU core and 16GB RAMs.

In the proposed hardware architecture, at the very beginning, the necessary pixels are loaded from BRAMs into PE array in 16 clock cycles. There are 1089 search locations because of the [-16, 16] search range. Additionally, there is 16 clock cycles latency due to pipelining architecture. Therefore 16+1089+16 = 1121 clock cycles are needed by the proposed hardware architecture to implement the algorithm proposed in [14]. Since the proposed hardware architecture can work at 150 MHz, therefore it can process 37 1280×720 full HD fps. It is worth noting in the Table 1 that hardware architecture proposed in [14] can process more fps compared to hardware architecture proposed in this paper but it consumes considerably more power. Furthermore, the architecture proposed in [14] cannot be synthesized for bigger block size such as 64×64 which is necessary for high efficiency video coding (HEVC).

4. CONCLUSION

In this paper, efficient hardware architecture is proposed for low complexity ME algorithm recently presented in [14]. Proposed hardware architecture is more efficient than the architecture proposed in [14] in terms of area and power consumption. The proposed architecture is also more suitable for bigger block sizes because of its simpler BRAM based memory hierarchy compared to the DFF based memory structure as in [14]. The architecture proposed in [14] cannot be synthesized for a block size of 64×64 which is necessary for high efficiency video coding (HEVC). On the other hand, the block size of 64×64can be processed by the architecture proposed in this paper where BRAM based implementation is performed

thanks to which the wire congestion in the architecture is reduced considerably. Utilization of LUTs in the proposed architecture is also lower compared to the architecture proposed in [14]. This is mainly because the reduced datapath complexity achieved by removal of the DFF based memory organization. Furthermore, fast search algorithm support is provided thanks to the ladder shaped memory scheme proposed in [2]. Thus, an early termination scheme is now possible to implement as a future work thanks to which higher frame rates may be achievable.

ACKNOWLEDGEMENTS

This work is supported by TUBİTAK under project number 115E921.

Table 1. *ME performance comparison of the hardware architectures*

	Proposed	[14] Recompiled	[7]	[16]	[17]
Bit depth	3	3	1	1	2
On chip memory	7168	0	24064	4096	NA
Area	7587 LUTs/ 2219 DFFs	8670 LUTs/ 7877 DFFs	1121LUTs NAs	3914 LUTs 2517 DFFs	5413 LUTs NA
Power	15.78 mW 50 MHz	17 mW 50 MHz	35.3 mW 50 MHz	NA	30.7 mW 50 MHz
Maximum frequency	150 MHz	243 MHz	218 MHz	192 MHz	275 MHz
Technology	FPGA 28 nm	FPGA 28 nm	FPGA 45nm	FPGA 65nm	FPGA 45nm
Search range	[-16 16]	[-16 16]	[-16,16]	[-16,16]	[-1,1] to [-16,16]
Search method	Spiral	Spiral	Full	Full	Spiral
Performance 720p@ #fps	37	61	NA	NA	NA
Performance 480i@ #fps	111	185	NA	NA	NA
Suitable for HEVC	Yes	No	Yes	Yes	Yes

Table 2. *Power consumption comparison (xc7a200tfbg676-2)*

Video Sequence: football Frame No: 09 Frame Size: 352×240							
Block locations		Motion Vectors		Power Consumption (mW) 20 ns/10 ns			
Vertical	Horizontal	hv_x	hv_y	Signals		Logic	
---	---	---	---	Proposed	[14]	Proposed	[14]

7	6	-3	-3	09/09	10/14	08/08	08/12
9	17	3	2	09/09	12/17	09/08	10/14
4	12	4	0	07/08	08/10	07/07	07/09
7	16	8	7	09/10	11/16	09/09	10/14
4	14	9	-7	07/08	08/12	07/07	07/10
2	6	9	9	07/07	06/10	06/06	05/08
13	16	-15	-11	09/10	12/19	09/08	11/16
4	20	14	-14	08/09	09/12	08/07	08/11
2	7	-12	12	07/07	06/10	07/06	05/08
-------		Average		8.00/8.55	9.11/13.3	7.78/7.33	7.89/11.3

Table 3. *Power analysıs results of the proposed hardware architecture (xc7z100ffv1156-2)*

Video Sequence: football Frame No: 09 Frame Size: 352×240					
Block locations		Motion Vector Position		Power Consumption (mW) 20 ns/10 ns	
Vertical	Horizontal	X Axis	Y Axis	Signals	Logic
7	6	-3	-3	09/09	08/08
9	17	3	2	09/10	09/09
4	12	4	0	07/08	07/07
7	16	8	7	10/10	09/09
4	14	9	-7	07/08	07/07
2	6	9	9	07/07	07/06
13	16	-15	-11	09/10	09/08
4	20	14	-14	08/08	08/07
2	7	-12	12	07/07	07/07
-------		Average		8.11/8.56	7.89/7.56

CONFLICT OF INTEREST

No conflict of interest was declared by the authors.

REFERENCES

[1] Kuhn, PM., Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation 1st ed., Norwell, MA, USA, (1999).

[2] Chen, T-C., Chen, Y-H., Tsai, S-F., Chien, S-Y. and Chen, L-G., "Fast Algorithm and Architecture Design of Low-Power Integer Motion Estimation for H.264/AVC", IEEE T CIRC SYST VID, 17: 568-577, (2007).

[3] Wei, C., Hui, H., Jiarong, T. and Hao, M., "A High-performance Reconfigurable VLSI Architecture for VBSME in H.264", IEEE T CONSUM ELECTR, 54: 1338-1345, (2008).

[4] Natarajan, B., Bhaskaran, V. and Konstantinides, K., "Low-complexity block-based motion estimation via one-bit transforms", IEEE T CIRC SYST VID, 7: 702-706, (1997).

[5] Ertürk, A. and Ertürk, S., "Two-Bit Transform for Binary Block Motion Estimation", IEEE T CIRC SYST VID, 15: 938-946, (2005).

[6] Urhan, O. and Ertürk, S., "Constrained one-bit transform for low-complexity block motion estimation", IEEE T CIRC SYST VID, 17: 478-482, (2007).

[7] Çelebi, A., Urhan O., Hamzaoğlu, I. and Ertürk, S., "Efficient hardware implementations of low bit depth motion estimation algorithms", IEEE SIGNAL PROC LET 16: 513-516, (2009).

[8] Akın, A., Doğan, Y. and Hamzaoğlu, I., "High Performance Hardware Architectures for One Bit Transform Based Motion Estimation", IEEE T CONSUM ELECTR, 55: 941-949, (2009).

[9] Çelebi, A., Akbulut, O., Urhan, O., Hamzaoğlu, I. and Ertürk, S., "An All Binary Sub-Pixel Motion Estimation Approach and its Hardware Architecture", IEEE Trans IEEE T CONSUM ELECTR, 54: 1928-1937, (2008).

[10] Çelebi, A., Akbulut, O., Urhan, O. and Ertürk, S., "Truncated Gray-Coded Bit-Plane Matching Based Motion Estimation and its Hardware Architecture", IEEE T CONSUM ELECTR, 55: 1530-1536, (2009).

[11] Chatterjee, SK., "Implementation of weighted constrained one-bit transformation based fast motion estimation", IEEE T CONSUM ELECTR, 58: 646-653, (2012).

[12] Çelebi, A., Lee, HJ. and Ertürk, S., "Bit plane matching based variable block size motion estimation method and its hardware architecture", IEEE T CONSUM ELECTR, 56: 1625-1633, (2010).

[13] Kuo, TY. and Wang, CH., "Fast local motion estimation and robust global motion decision for digital image stabilization", Intelligent Information Hiding and Multimedia Signal Processing, IIHMSP '08 International Conference, Harbin-China, 442-445, (2008).

[14] Yavuz, S., Çelebi, A., Aslam, M. and Urhan, O., "Selective Gray-Coded Bit-Plane Based Low-Complexity Motion Estimation and its Hardware Architecture", IEEE T CONSUM ELECTR, 62: 76-84, (2016).

[15] Tuan, JC., Chang, TS. and Jen, CW., "On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture", IEEE T CIRC SYST VID, 12: 61-72, (2002).

[16] Akın, A., Sayılar, G. and Hamzaoğlu, I., "High performance hardware architectures for one bit transform based single and multiple reference frame motion estimation", IEEE T CONSUM ELECTR, 56: 1144-1152, (2010).

[17] Çelebi, A. and Urhan, O., "High performance hardware architecture for constrained one-bit transform based motion estimation", Signal Processing Conference, Barcelona-Spain, 2151-2155, (2011).