VLSI Implementation of High Performance Optimized Architecture for Video Coding Standards

S. Rukmani Devi[1], P. Rangarajan[2], and J. Raja Paul Perinbam[3]

[1)] Electronics and Communication Department, R.M.K. Engineering College, Chennai -601206, India, e-mail: drd.ece@rmkec.ac.in

[2)] Electrical and Electronics Department, R.M.D. Engineering College, Chennai-601206, India, e-mail: hodeee@rmd.ac.in

[3)] Electronics and Communication Department, Karpaga Vinayaka College of Engineering and Technology, Chennai-601206, India, e-mail: jrpp@annauniv.edu

Abstract: This study presents a fast search algorithm and its Very Large Scale Integration (VLSI) design to implement an Enhanced Diamond Search (EDS) of Block-based Motion estimation for video compression systems. The proposed algorithm reduces the number of search points with a slight increase in average Sum-of-Absolute Difference (SAD) per pixel and significant reduction in Peak Signal-to-Noise Ratio (PSNR) than the Full search (FS). The speed improvement ratio is 11.26%. A novel Motion Estimation (ME) algorithm is proposed in this study that exhibits some properties with potential to facilitate optimized VLSI implementation and to achieve high performance for real video sequences. The main characteristic of this proposed architecture is possessing of only five processing elements (PE) that are used to calculate minimum SAD by using efficient comparators and compressors. Simulation results indicate that our proposed architecture employs low power and processes the video data with high speed than existing architectures without significant change in the video quality.

Keywords: Motion Estimation; VLSI; Diamond search; video compression; PSNR; SAD

1 Introduction

Video compression standards is developed by international organizations such as ISO/IEC and ITU-T. ISO/IEC MPEG standard includes MPEG-1, MPEG-2, MPEG-4 Part-2, MPEG-4 Part 10 (AVC), MPEG-7, MPEG-21 and M-JPEG. ITU-T VCEG standard includes H.26x series (H.261, H.263 and H.264) [1]. In attaining the video coding standards, Motion Estimation (ME) plays a vital role to achieve significant compression by exploiting temporal redundancy existing in a video sequence. ME is the most computationally intensive functions of the entire video coding process. Of several algorithms available for block matching motion estimation, full search algorithm identifies the best match in the entire search window by computation of SAD at each location. For a search window of size +/- W pixels, the number of search locations is $(2W+1)^2$[2]. For a search window of 31 x 31 and a block size of 16 x 16, a total of 961 locations will be searched to obtain a best match with a minimum SAD value. This results in significant computational complexity and consumes up to 80% of the computing power of the encoder. To reduce the computational complexity, numerous optimized search algorithm are available as follows: Three Step Search (TSS), New Three Step Search (NTSS), Four Step Search (FSS), Block Based Gradient Descent Search (BBGDS), Diamond Search (DS), Hexagon–Based Search (HEXBS), Adaptive Rood Pattern Search (ARPS), Cross Diamond Search (CDS) [3-12]. The TSS reduces the number of computations by using a coarse-to-fine search strategy. The other algorithms mentioned above reduces the number of computations in relation to TSS by using a center-biased motion vector distribution characteristics. In real world video sequences, the motion of blocks are as stationary or quasi-stationary. If blocks are stationary (about 40%-60%), the corresponding Motion Vectors (MVs) are located at the search center and if the blocks are quasi-stationary (30% -40%), the corresponding MVs are enclosed in a window of size ±2 pixel distance around the center [13-14].

In this paper, we propose an Enhanced Diamond Search (EDS) algorithm for fast block motion estimation algorithm. After analyzing, the characteristics of the MVs, the search pattern of Large Cross Diamond Pattern (LCDP) consists of five search points with pixel distances of ±2 and Small Cross Diamond Pattern (SCDP) consists of four search points with pixel distance of ±1 based on the origin. This proposed algorithm achieves fewer search points and better Peak Signal-to-Noise Ratio (PSNR) than DS, CDS and other fast search algorithms.

The real time processing of video signals requires tremendous computational capabilities that can only be achieved cost effectively by using VLSI. The usefulness of motion estimation algorithms strongly depends on the feasibility and effectiveness of its VLSI implementation. We propose low cost optimized high performance architecture for EDS algorithm used for low bit rate applications.

This paper is organized as follows: Section 2 discusses about proposed algorithm, Section 3 deals with proposed hardware architecture Section 4 describes about synthesis results and conclusion given in the Section 5.

2 Enhanced Diamond Search Algorithm

The EDS algorithm has two cross diamond patterns such as a large cross diamond pattern (LCDP) and small cross diamond pattern (SCDP) as shown in Figure 1(a) and Figure 1(b). It consists of five candidate search points suitable for exploiting the center biased characteristic of motion vector distribution. Figure 2 shows the position of the diamond, with respect to the previous position, for the next search. At subsequent steps, three or two new candidate search points to be evaluated with the maximum overlapping region are required to minimize the number of search points. To obtain the minimum SAD, the final search step with four new candidate search points is evaluated.

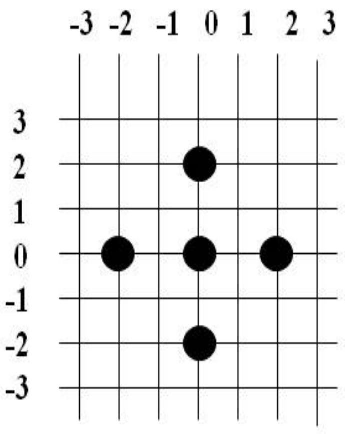


Figure 1 (a)
Large Cross Diamond Pattern (LCDP)

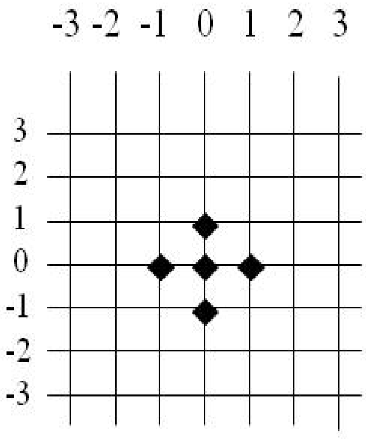


Figure 1(b)
Small Cross Diamond Pattern (SCDP)

Figure 1
The EDS patterns

In order to describe the EDS algorithm, a sum of absolute difference (SAD) obtained by employing the following formula[8]

$$SAD(i,j) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} |CB(m,n) - RB(m+i, n+j)| \quad (1)$$

Where N is the block size & CB and RB are the pixel values in the current block and the reference block respectively. PSNR characterizes the motion compensated image created by predicting motion vectors and blocks from the reference frame using the following formula [3]

$$\text{PSNR} = 10 \log_{10} \{(2^b - 1)^2/\text{MSE}\} \quad (2)$$

Where b is the number of bits per pixel and MSE is Mean Square Error.

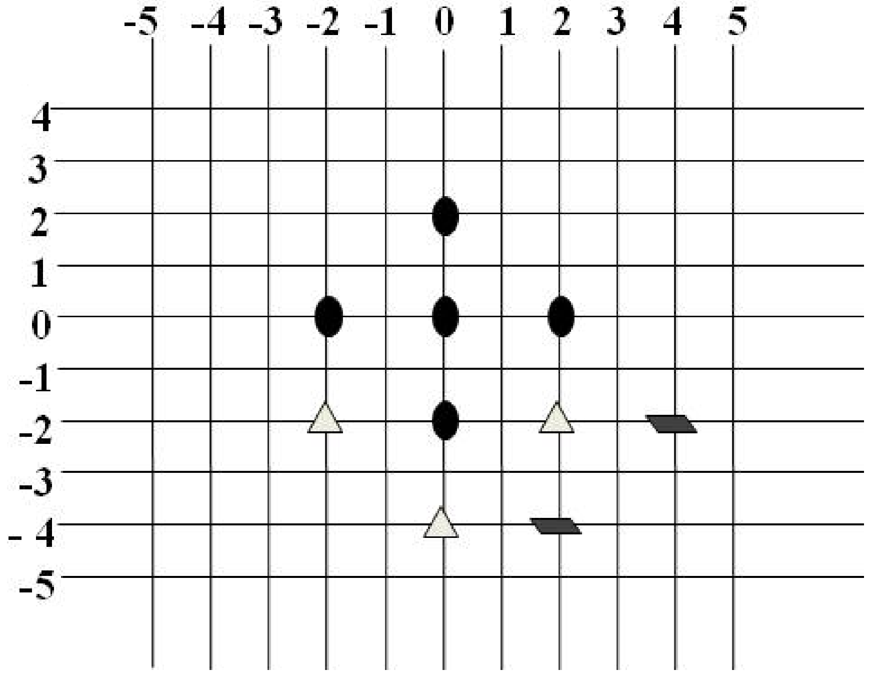


Figure 2
Search path example

The EDS algorithm is summarized as follows:

Step 1. The LCDP is placed at (0, 0) the center of the search window. The SAD is calculated for each of the 5 candidate search points .If the minimum SAD is found to be at the center (a, a) of the LCDP, jump to step 4.

Step 2. If the minimum SAD in the previous search step is located at any one of the vertices i.e. (a±2, a) and (a, a±2), then proceed with step 3.

Step 3. Obtain minimum SAD by repositioning the LCDP considering three and two new candidate search points with a pixel distance of ±2. If the new minimum SAD is still at the center of the newly formed LCDP, proceed to step 4, otherwise continue with step 3. A candidate search point that extends beyond the search window is ignored.

Step 4. A new SCDP is formed, if the minimum SAD is in the center of LCDP. Add four new search points with a pixel distance of ±1 to identify the new minimum SAD. If the minimum SAD is at the center of SCDP, the algorithm stops and corresponding SAD value is considered as the final value of MV. The flowchart of this algorithm is shown in Figure 3.

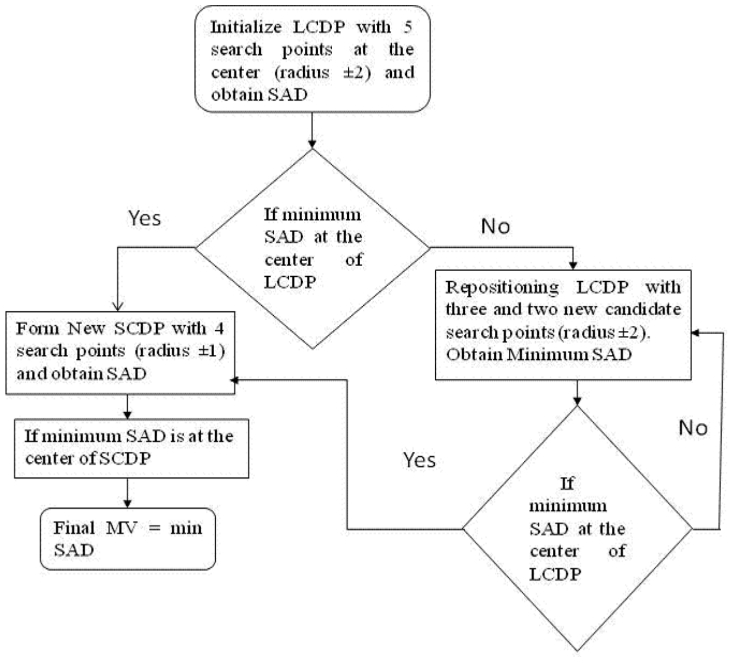

Initialize LCDP with 5 search points at the center (radius ±2) and obtain SAD
If minimum SAD at the center of LCDP
Yes
No
Form New SCDP with 4 search points (radius ±1) and obtain SAD
Repositioning LCDP with three and two new candidate search points (radius ±2). Obtain Minimum SAD
If minimum SAD is at the center of SCDP
Yes
No
If minimum SAD at the center of LCDP
Final MV = min SAD

Figure 3
Flowchart of the EDS algorithm

2.1 Software Analysis

A software analysis was performed to evaluate the average number of search points per block, average SAD per pixel and PSNR using MATLAB and results are compared for two different video sequences as given in the Table 1. The search area employed was 32 x 32 pixels with block size of 16 x 16 pixels. The algorithm was applied to the first 30 frames of CIF video sequences with a resolution of 352 x 288 pixels.

Table 1
Comparison of DS and proposed algorithm for two video sequences

Sequences	DS			**Proposed**		
	Average No. of search points / Block	Average SAD / Pixel	Peak Signal-to-Noise Ratio (PSNR)	**Average No. of search points / Block**	**Average SAD / Pixel**	**Peak Signal-to-Noise Ratio (PSNR)**
Foreman	13.99	5.920	33.591	**10.30**	**5.996**	**33.615**
Mobile	17.51	11.734	32.582	**11.62**	**11.957**	**32.699**

3 Proposed Hardware Architecture

The proposed architecture consists of current block memory, reference block memory, data-fetch unit, data-fetch initializer, Processing Element (PE) array, comparator, PE array Enabler and Timing and control unit as shown in Figure 4. To start Motion Estimation (ME) process, it is necessary to fill the reference block memory and the current memory block. The reference block memory is 32 x 32 bytes memory, which can store all data to perform SAD operations and the final refinement. The current block memory is 16 x 16 bytes and it contains the block being processed. The data-fetch initializer initializes the data-fetch unit to fetch reference pixels and current pixels from the memory, row and column address of the block at the center of the reference frame and type of fetching i.e. pixels of horizontal or vertical direction .It also calculates starting address of the values to be fetched, number of search points to be fetched at a time and number of cycles needed to fetch data input. After initializing, the data-fetch unit obtains the pixel values from the current and reference block. The pixel values are applied to the PE array. A PE array with 5 processing elements is used to calculate SAD between the current block and reference block. Totally five SAD values are generated by the PE array and these are compared by the comparator to evaluate minimum SAD or best motion vector and position of the motion vector. The process is repeated until the comparator finds the lowest SAD in the center of the EDS. Low power can be achieved by using the PE array enabler. This block is used to enable appropriate processing element based on the search path. The timing and control unit is a finite state machine, which is used to control all the blocks presented in the proposed architecture.

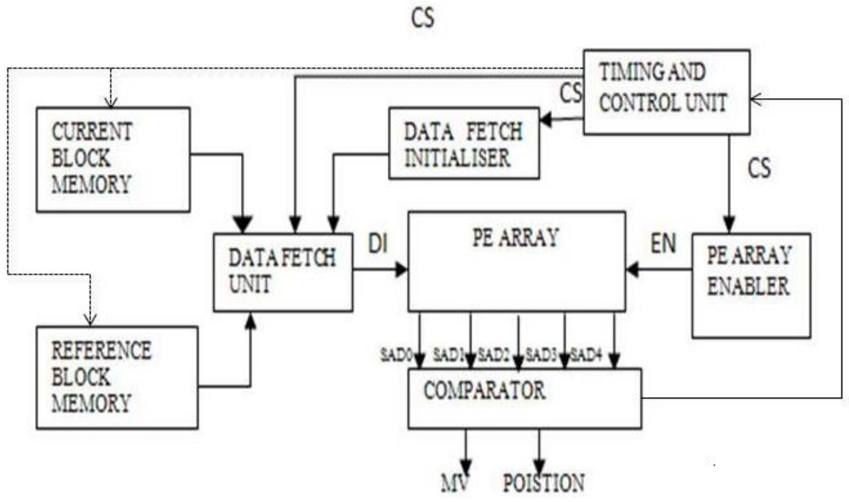


Figure 4
Proposed Optimized Architecture of the EDS algorithm

3.1 Processing Element

The PE array was designed with five processing elements and four stage pipeline for better performance. It consists of an absolute difference unit, adder array and accumulator as shown in Figure 5. The internal structure of the PE is composed of a large number of addition operations to calculate the SAD. These operations can be performed effectively by using 5-2 and 3-2 adder compressors [15-19] as shown Figure 8(a).

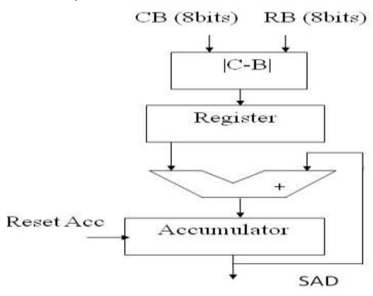


Figure 5
Processing Element

The absolute difference unit is used to evaluate the absolute differences between two pixels. It consists of eight 1-bit comparators (1-bit CMP), seven 2-bit comparators (2-bit CMP) and two XOR arrays [15-19] as shown in Figure 6. The 1-bit CMP is formed by a XOR gate and two AND gates. It produces three output signals: they are less than (LE), Equal to (NE) and greater than (LG), as shown in Figure 7 (a). The 2-bit comparator is formed by using two 1-bit CMP as shown in Figure 7 (b).

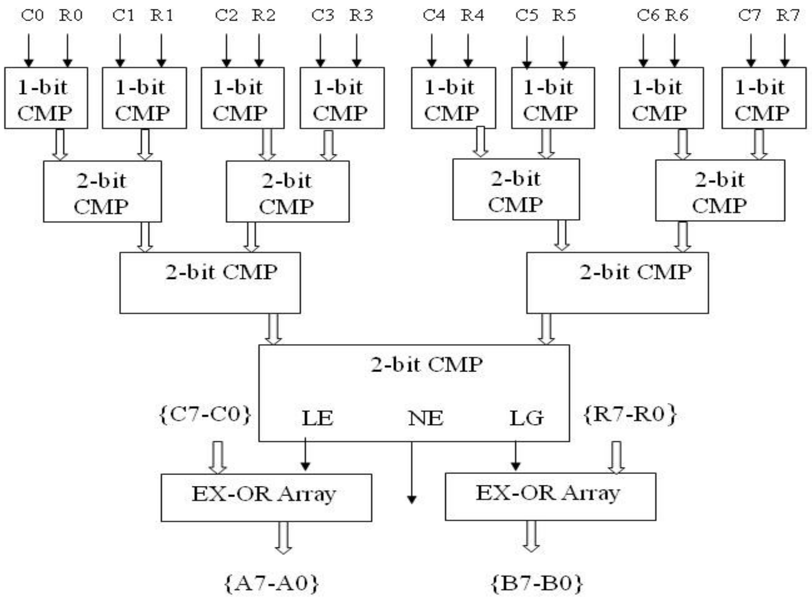
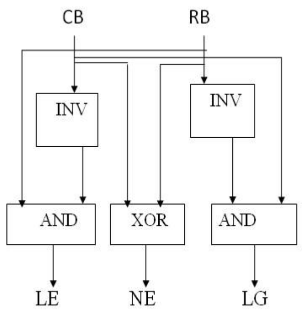


Figure 6
Absolute Difference Unit (Proposed by LiYufei et.al)



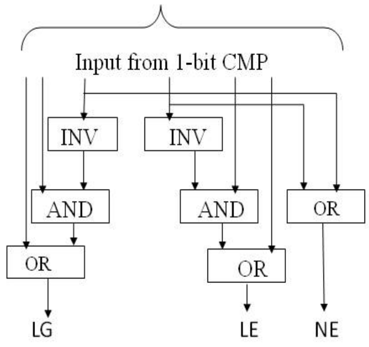


Figure 7 (a)
1-bit Comparator

Figure 7 (b)
2-bit Comparator

Processing element (PE) performs a large number of additions that can be optimized by using 5-2 and 3-2 compressor for an 8-bit number as shown in Figure 8. The internal structure of 5-2 and 3-2 compressors are shown in Figure 9(a) and Figure 9(b).

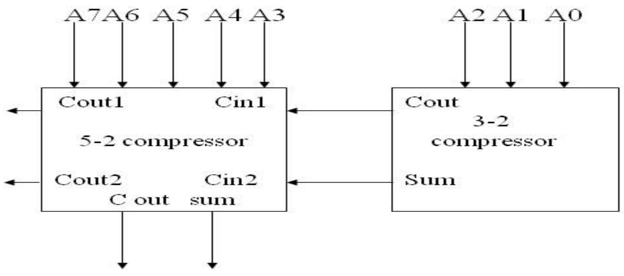
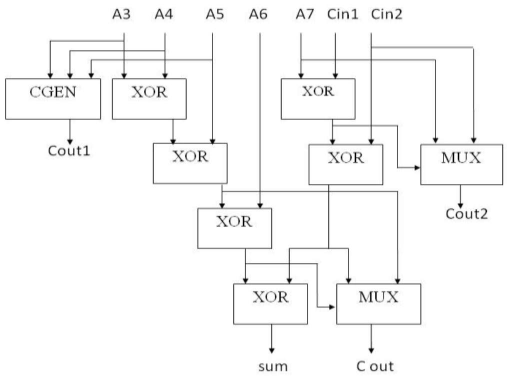


Figure 8

5-2 Compressor and 3-2 Compressor



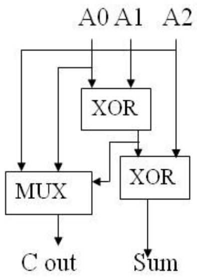


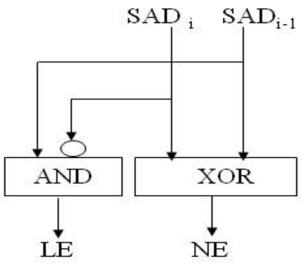
Figure 9 (a)

Internal structure of 5-2 compressor

Figure 9 (b)

Internal structure of 3-2 compressor

Depending on the search path, the PE generates SAD values. These values are compared by using comparator (as shown in figure 4) and minimum SAD is obtained. For optimization purposes, the comparator is constructed using 1-bit less comparator (1-bit LCMP) and 2-bit less comparator (2-bit LCMP) as shown in Figure 10(a) and Figure 10(b).



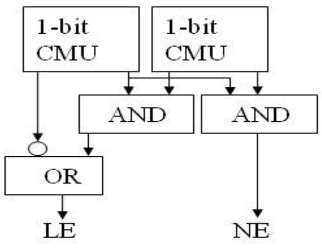


Figure 10 (a)

1-bit LCMP

Figure10(b)

2-bit LCMP

3.2 Timing and Control Unit

The timing and control unit generates a control signal (CS) to control all the blocks present in the proposed architecture (refer Figure 4). The control unit is a Finite State Machine (FSM) that controls the operation of each unit based on the search path. The flow remains to be linear owing to simpler control of the architecture. Table 2 explains the operation of the control unit.

Table 2
Timing and Control signal

State	State name	Description
S0	CLEAR	Clears PE, ACC, DF unit, DF initialize, PE array enabler & waits for a clear signal to move to the next state.
S1	EXTERNAL INPUT STATE (DI)	Receives and stores the external inputs
S2	HORIZONTAL PHASE STATE	Selects search type and starts horizontal computation
S3	SAD COMPUTATION STATE	Starts the sad computation by making enable (EN) =1
S4	COMPARATOR INITIALISE STATE	Comparison of the SAD values for horizontal pixels to find minimum SAD
S5	HORIZONTAL COMPLETE STATE	Clears PE, ACC, and DF unit PE array enabler.
S6	VERTICAL PHASE STATE	Selects search type and starts vertical computation
S7	SAD COMPUTATION STATE	Starts the sad computation for vertical by making enable (EN) =1
S8	COMPARATOR INITIALISE STATE	Comparison of the SAD values for vertical pixels to find minimum SAD
S9	FINISH	If minimum SAD is obtained go to the next block

4 Synthesis Results

The proposed architecture was implemented in Verilog HDL using Xilinx 12.2 in Spartan 6 SP605 FPGA LX45T kit and output was analyzed using Chip Scope Pro. Table 3 presents the synthesis results of this proposed algorithm for two different devices. The device hardware utilization is small, since only 3214 look-up-tables (FPGA LUTs) are used. The performance results presented in Table 3 take into account CIF (352 x 288 pixels) videos. This hardware consumes 1711 slices, which is 27% of all the slices of an xc6slx45Tfgg484-3 with maximum frequency of 397.844 MHz and power consumption of 120 mw with minimum latency. Table 4 gives the comparison of the proposed architecture with other architectures.

Table 3
Synthesis Results

Parameters	**EDS Results**	
	(xc6slx45Tfgg484-3)	(2v40cs144-5)
No. of Slices	1711 (27%)	87(33%)
No. of Flip flops	384 (3%)	30(5%)
No. of 4 I/P LUTs	3214 (26%)	156(30%)
Minimum Period	2.514ns	4.227ns
Maximum frequency (MHz)	397.84	236.574

Table 4 compares various parameters between the proposed and existing diamond search algorithms.

Table 4
Performance Comparison of Proposed with other architectures

Parameters	QSDS [15]	SDS[20]	**Proposed**
No. of Slices	2007	1968	**1711**
No. of Flip flops	2086	2020	**384**
No. of 4 I/P LUTs	3610	3541	**3214**
Minimum Period	4.688 ns	-	**2.514 ns**
Maximum frequency (MHz)	213.3	185.7	**397.84**
Number of PE	9	9	**5**
Search range	64 x 64	64 x 64	**32 x 32**

Conclusion

The enhanced diamond search algorithm displayed better performances as compared to full search and other fast search algorithms. This algorithm performs a less number of search points and 0.024 dB increase in PSNR for medium motion sequence (Foreman) and 0.177 dB increase in object translation sequences (Mobile) with 11.26% speed improvement ratio than other algorithms. In this study, a VLSI architecture was developed for this algorithm, which uses only five processing elements to maintain 30 f/s with an operating frequency of 397 MHz. Totally, 32 clock cycles per block needed to obtain a SAD value. To improve the performance of this architecture, optimized PE was incorporated. This algorithm is developed for fixed block size motion estimation and was implemented successfully in FPGA device. This work could be extended to variable block size motion estimation for an H.264 AVC standard.

References

[1] Bhaskaran and K. Konstantinides, Image and Video Compression Standards Algorithms and Architectures, Kluwer Academic, Boston, Mass, USA, 1999

[2] S.-Y. Huang. Chuan-Yu Cho ; Jia-Shung Wang "Adaptive Fast Block-Matching Algorithm of Switching Search Patterns for Sequences With Wide-Range Motion Content", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 15, No. 11, pp. 1383-1384, November 2005

[3] Shiping Zhu, Jun Tian, Xiaodong Shen, Kamel Belloulata," A New Cross-Diamond Search Algorithm for Fast Block Motion Estimation", pp. 1581-1584, ICIP-2009

[4] www.cwaip.nus.edu.sg

[5] Sanchez, Gustavo, Felipe Sampaio, Marcelo Porto, Sergio Bampi, and Luciano Agostini. "DMPDS: A Fast Motion Estimation Algorithm Targeting High Resolution Videos and Its FPGA Implementation", International Journal of Reconfigurable Computing, pp. 1-12, October 2012

[6] "IEEE-ICDCS conference proceeding", 2012 International Conference on Devices Circuits and Systems (ICDCS), 03/2012

[7] www.ee.cityu.edu.hk

[8] Reeba Korah, Sankaralingam and M. J. R. P. Perinbam, "Motion Estimation with Candidate Block and Pixel Subsampling Algorithm", IEEE International Workshop on Imaging Systems and Techniques, pp. 130-133, May 2005

[9] Porto. M, Agostini, L. , Bampi, S. and Susin, A. "A High Throughput and Low Cost Diamond Search Architecture for HDTV Motion Estimation", 2008 IEEE International Conference on Multimedia and Expo, pp 1033-1036, June 2008

[10] www.ee.vt.edu

[11] Su-Bong Hong, Hyoseok Lee, Geun-Young Chun, Hyunki Baik and Myong-Soon Park, "FCBHS: a Fast Center-biased Hybrid Search Algorithm for Fast Block Motion Estimation", Proceedings International Conference on Information Technology Coding and Computing ITCC, pp. 254-259, Feb 2002

[12] Yong Ding, Xiao-Lang Yan, "Parallel Architecture of Motion Estimation for Video Format Conversion with Center-biased Diamond Search", International Conference on Information Engineering and Computer Science, pp. 1-4, Dec 2009

[13] Huang, Hui-Yu, and Shih-Hsu Chang. "Block Motion Estimation Based on Search Pattern and Predictor", IEEE Symposium on Computational Intelligence For Multimedia Signal and Vision Processing, pp. 47-51, April 2011

[14] Mahid Asefi, Mohamed-yahia Dabbagh. "Adaptive Video Motion Estimation Algorithm via Estimation of Motion Length Distribution and

Bayesian Classification", IEEE International Symposium on Signal Processing and Information Technology, pp. 807-810, August 2006

[15] Marcelo Porto, André Silva, Sergio Almeida Eduardo DA Costa, Sergio Bampi," Motion Estimation Architecture Using Efficient Adder-Compressors for HDTV Video Coding", Journal Integrated Circuits and Systems, Vol. 5, No. 1, pp. 78-88, 2010

[16] LiYufei , Feng Xiubo and Wang Qin," A High-Performance Low Cost SAD Architecture for Video Coding', IEEE Transactions on Consumer Electronics, pp. 535-541, Vol. 53, No. 2, May 2007

[17] Jarno, Vanne,Eero Aho, Timo D. Hämäläinen, and Kimmo Kuusilinna, "A High-Performance Sum of Absolute Difference Implementation Motion Estimation, IEEE Transactions on Circuits and Systems for Video Technology, pp. 876-883, Vol. 16, No. 7, 2006

[18] Chip-Hong Chang, Jiangmin GU, and Mingyan Zhang," Ultra Low-Voltage Low-Power CMOS 4- 2 and 5-2 Compressors for Fast Arithmetic Circuits", IEEE Transactions on Circuits and Systems—I: Regular Papers, pp. 1985-1997, Vol. 51, No. 10, October 2004

[19] Meihua Gu, Ningmei Yu, Lei Zhu, Wenhua Jia," High Throughput and Cost Efficient VLSI Architecture of Integer Motion Estimation For H.264/AVC", Journal of Computational Information Systems, pp. 1310-1318, Vol. 7, No. 4, April 2011

[20] Marcelo Porto, Luciano Agostini, Sergio Bampi, AltamiroSusin," A High throughput And Low Cost Diamond Search Architecture For HDTV Motion Estimation", pp. 1033-1036, ICME 2008