International Journal of Engineering & Technology, 7 (4.10) (2018) 1075-1078

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Hardware Implementation of Diamond Search Algorithm for Motion Estimation

Prayline Rajabai C, Harish J, Sivanantham S*

School of Electronics Engineering,
Vellore Institute of Technology, Vellore, Tamil Nadu-632014, India.
**Corresponding author E-mail: ssivanantham@vit.ac.in*

Abstract

Motion estimation one of the advanced technique adapted in the industry for video coding and implemented in various applications. This work is focused on the efficient hardware implementation of the diamond search algorithm architecture. Among all the other Fast Search algorithms like three-step search, new three-step search, four-step search and diamond search (DS), the diamond search algorithm maintains the diamond shape search pattern and it gives faster search pattern and minimum absolute difference. When compared with the original diamond search (DS) algorithm, this modified diamond search algorithm requires less area and power maintaining the same performance. Other than the architectural level changes the low power synthesis is done and the results show that this design can be implemented effectively for an application that require fast search with low power requirements like IoT and sensor devices. The design has been implemented using Verilog HDL, synthesized using Synopsys DC compiler using 90nm technology.

Keywords: *Motion Estimation, Diamond Search Algorithm, Low Power, Video Coding, Hardware Architectures, IoT and Sensor devices.*

1. Introduction

Motion estimation is used for temporal prediction in video coding. In most of the cases, the pixel information in the consecutive frames will be similar and except the changes introduced by the objects changing within the frame. Motion estimation has proven to be effective in utilizing the temporal redundancy of video sequences and therefore forms a central part of all the hybrid video compression standards. Objects that are seen at a particular area on one frame move within the successive frames to form the corresponding objects on the subsequent frame. Temporal redundancy between consecutive frames can be exploited in many ways. One of the simpler methods is frame differencing. This strategy proves that as the average motion is small and hence by compressing the pixel difference between two frames instead of compressing the full frame we can achieve high compression ratio [1]. Two categories of ME algorithms are full search algorithms and fast search algorithms. These algorithms can be either lossy or lossless. Generally, full search algorithms are lossless and fast search algorithms are lossy [2]. These algorithms can be block matching algorithms or pixel matching algorithms. All these algorithms operate on the reduction of the search points and improve the throughput. Among all the different algorithms, block matching algorithms such as full search block matching algorithm (FSBMA), Diamond search algorithm are a few popular techniques [3]. These algorithms implemented in hardware should be highly efficient to suit for real-time applications [4].

Motion estimation and motion compensated prediction is by far the motion estimation and widely used technique for achieving the high levels of compression that are typified in modern video compression standards In this techniques a scene of a frame is arbitrarily divided into Macroblocks [5]. Each Macroblock is closely associated with some pixel values. Successive elimination algorithm attracts lots of interest because it can provide the same image quality as FSBMA but with less computation [6].

Motion Compensation is defined as during reconstruction, the reference frame is used to predict the current frame using the motion vectors. During the motion compensation, the macroblock in the reference frame that is referenced by the motion vector is copied into the reconstructed frame.

Block Matching algorithm is one of the popular technique to reduce the temporal redundancy. MPEG-1, MPEG-2, CCITT, H.261, H.263, and H.264 are some of the coding standards which are widely using the Block Matching Algorithm [7]. Object tracking is also a vital task in computer vision which uses the Motion Estimation technique [8].

To estimate 2D motion recursively on a pixel basis pixel recursive motion estimation algorithms are used. Pixel recursive techniques are used to determine the motion field and also used for adequately computing the gradient. Fast search block matching algorithm is used primarily to reduce the complications in the performance of the system [8]. However, even though it improves the speed of the operation, it affects the quality of reconstruction of the frame. The practical usage of the fast search block matching algorithm will balance the speed and reconstruction of the frame. In comparison with the full search algorithm, another popular method for fast search is a three-step algorithm. It is robust and optimal in performance by assuming the initial step size and the best match is found by comparison of eight points around the center at a distance of step size. The best match at the end of the first iteration is considered as the center for the next iteration with step size reduced by a factor of 2 [9]. This process is repeated until we get the best match as the center.

Video compression has played a prominent role in storing the video data as well as in the transmission of the video data. Video conference, remote monitoring, and video phones are some of the examples which use the video compression a lot [10,11,12,13]. Visual information is one of the new advents of the technology which needs to be compressed because of the enormous consumption of the signal bandwidths.

2. Related Works

As the importance of fast motion estimation is growing, a lot of contributions are done for the betterment of performance and hardware [6, 14]. This section mainly explains some of the contributions which describe the implementation of the motion estimation algorithm. This work has mainly contributed the efforts in reducing the number of search points for finding the best possible motion vector. The procedure involved is as follows, at first MDS does the Small diamond search pattern in the initial search to get the minimum distortion point. In the second step if MBD point is not located in the center then comes the Simplified Large Diamond search pattern(SLDSP) into the picture. If the MBD point is not in the circular area, then SLDSP will be repeated until the MDB point becomes the center. Then as a part of the final step, the SDSP will be formed. The main idea of this work predicting the motion vector by employing the initial SDSP, so that the number of search points got reduced.

It is clearly explained in [6], that the computational process changes for implementing the systolic array processor. The basic idea behind this paper is to map the motion estimation to the parallel processing architecture, i.e. by restructuring the SAD equation to achieve the pipelined parallel processing. In this way they better way of mapping the motion estimation to systolic array architectures. From the Fast Diamond search algorithm based on the conventional algorithm presented in [3], it is clearly understood their main contribution is to reduce the computations in finding the best possible motion vector. The main idea is to divide the conventional diamond search algorithm into two algorithms, first is to reduce the internal stop search which reduces the internal redundant SAD and second is to External stop search which reduces unnecessary computations by skipping the all the not needed and irrelevant blocks in the search algorithm.

Thus from the literature survey, it is clear that many works have been made a lot to contribute to the reduction of the number of search points and computations which directly impacts the performance which is appreciated. Now coming to the current work, this paper mainly concentrates on how the best we can implement the algorithm in terms of Hardware by maintaining the same performance or even better.

3. Proposed Architecture for Diamond Search Algorithm

In this work a novel architecture to perform the diamond search computation is implemented. It comprises of a Large Diamond block (LDB), add5 point block (A5B), add3 point (A3B), Small diamond block (SDB) and state machine controller block (SMB). All these units have the sole purpose of calculating the minimum distance from the reference frame and current frame. The Large Diamond block (LDB) consists of Distance Calculation block (DCB), Serial to parallel register and a comparator. The LDB is the main arithmetic unit which performs the exhaustive search operation to calculate the minimum distance between the frames. The add5 point and add3 point blocks are responsible for modifying the pixel locations of the present frame. The purpose of the Current RAM is to store the pixel coordinates of all the nine positions of the present frame and the reference RAM stores the reference pixel coordinates of all the nine positions of the reference frame. The Small Diamond Block (SDB) generates a small Diamond shape when the two frames coincide with their locations. Fig. 1 shows the architecture of the large diamond search algorithm which has a State Machine (FSM) which governs the operation of all submodules.

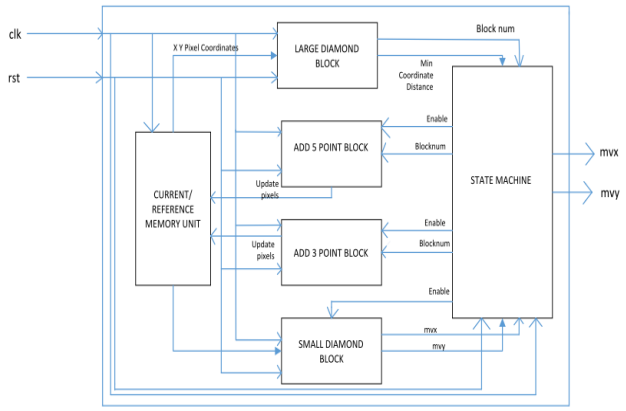

Fig. 1: Architecture of Large Diamond Search Algorithm.

4. Methodology

The Minimum Distance Calculation is done in Large Diamond Block (LDB). The Coordinates of both the frames are available in the respective rams. One of the Novel features of this project is that a single memory unit is accessible to all the blocks in spite of having separate memory provision for every module [8]. The pixel locations of the current and reference frame are obtained in DCB from memory. The distance between the coordinates is calculated here. These values are further passed to a serial to parallel register. After the SIPO is full, all the stored values are moved to the comparator in which the minimum distance is calculated. The block which has the minimum distance is also figured out in the LDB.

Once the Minimum distance along with its block is computed the state machine decides to invoke either the ADD5 or ADD3 block. If the minimum block is of an even number, the ADD3 point module is enabled, otherwise, ADD5 point is enabled. The ADD3 module checks the block number of the present frame and changes the overall coordinates thereafter. Below description shows the operation of the ADD3 point block.

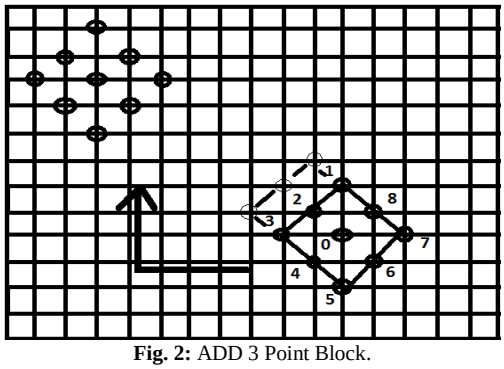

Fig. 2: ADD 3 Point Block.

From Fig. 2 the minimum block number is observed to be 2. Thereby it is necessary to shift the present frame one degree left in the x-axis and one degree above the y-axis. The ADD3 point block has to update all the pixel coordinates which are present in the current ram. ADD5 point block behaves in a similar manner but it deals with odd number blocks.

From Fig. 3 the minimum block appears to be 1. Thereby it is necessary to shift the present frame one degree above in the y-axis

keeping the x-axis same. The small diamond block (SDB) is triggered by the state machine when the minimum distance calculated by the LDB appears to be 0. The SDB endeavors to create a small diamond pattern after collecting the pixel coordinates of the central location of the current frame. The motion vectors thus obtained are nothing but the central coordinates of the current frame overlapping the reference frame. The motion vectors are finally passed on to the state machine which is given as an output.

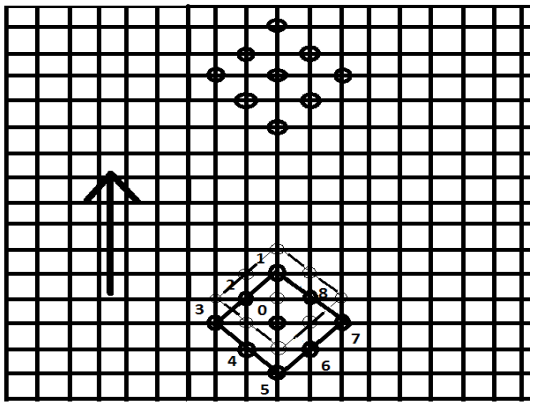

Fig. 3: ADD 5 Point Block.

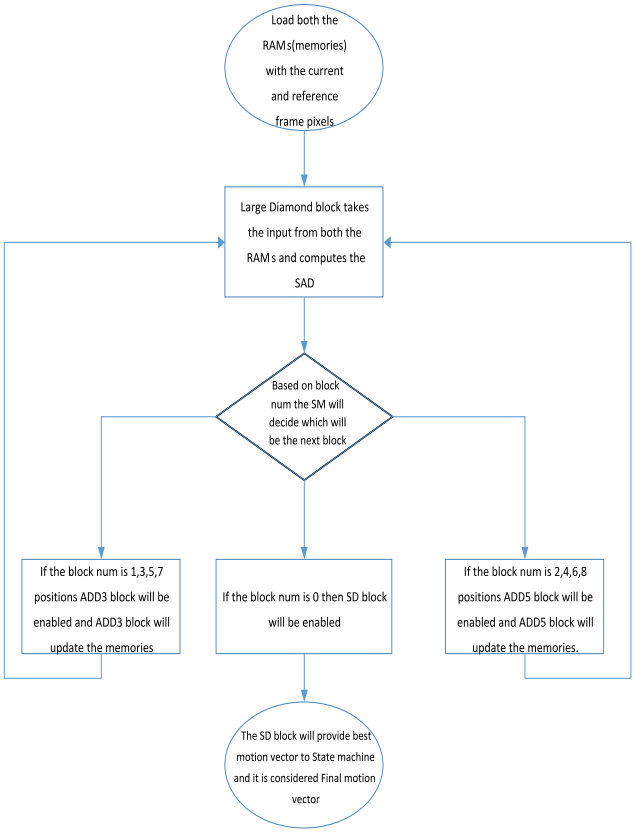

Fig. 4: Flow Chart

Fig. 4 explains the complete flow in implementing the Hardware efficient diamond search algorithm. Initially, both the current RAM and the reference RAM are loaded with the current frame pixels and reference frame pixels respectively. The large diamond block takes the input from both the memories where the current frame and reference frame pixels are stored. The Large Diamond block is having a logic of finding the minimum SAD and gives the output as block number to State Machine controller.

Based on the block number generated from the LDB, the State Machine controller will decide which block among SD, A3B and A5B have to be enabled. Based on the LDSP pattern it is configured that if block number is odd like 1, 3, 5, 7, then ADD3 point will be activated. If the block number is even like 2, 4, 6, 8, then ADD5 point will be activated else if it is zero then the small diamond block will be activated. Hence only anyone or the ADD5 point block, ADD3 point block or small diamond block will have the logic of updating the pixels locations to the memory. Hence based on the generated block number the computational complexity is reduced. This brought the novelty in the architecture by accommodating the SAD computation logic to only a large diamond block.

5. Large Diamond Block

Fig. 5 shows the internal architecture of the Large Diamond Block. The Large Diamond Block has the i) current RAM, ii) reference RAM, iii) PE block, iv) SIPO block and v) the comparator block. The Functionality of this block can be explained as follows. In Step I both the current and reference frame pixel locations are stored in their respective RAM. The Processing element (PE) computes the Sum of absolute difference calculation for every respective pixel location. In Step II the Serial input parallel output register (SIPO) takes each SAD from the PE and registers the data. Once for all the 9 locations is done during Step III all the 9 SAD's will be given as an input to the Comparator which computes min SAD among all the nine and its block number. This is given as input for State Machine Controller.

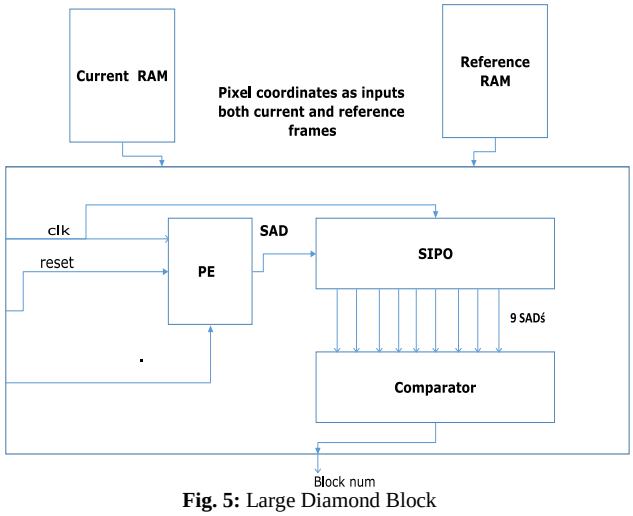

Fig. 5: Large Diamond Block

The processing element computes the cost function. The design of our processing element is in such a way that it can able to compute the cost function of the two 2D arrays of the current and reference search point's value in one clock pulse. Hence, it reduces computation time. The PE performs a SAD operation of the two search points coming from current and reference images. The SAD computation involves three main operations which are subtraction, absolute difference, and addition.

The SIPO block shown in Fig. 5, collects the output of the processing element serially and then converts it into parallel output so as to have nine SAD values in parallel (each one for each point). The SAD values are available at the output of the SIPO after nine clocks for a large diamond block. These SAD are given as input to the comparator block. The comparator compares all the SAD input values and gives the minimum SAD as the output denoted as the block number.

6. Results

The complete design is described and implemented using Verilog HDL and is simulated for functional verification using Questa

simulator. The gate level netlist has been generated using the Synopsys DC Compiler with the 90nm CMOS technology library. Low power synthesis is also done as part of exploring upf writing for the design. The synthesis results are shown in Table I.

Table 1: Synthesis Results

Design	Normal synthesis	Low power synthesis
Total dynamic power	1uW	800.1337 nW
Cell Leakage Power (mW)	36.1547 mW	2.739uW
Total Power(mW)	6.2465 mW	3.167 uW
Total area (sq.µm)	9448.85	1894.54

From the Table 1, it is observed that the area is reduced by three times which is very crucial for the applications which require lesser area. In addition, the 50% of power has been reduced which is essential for low power application like IoT or smart devices.

7. Conclusion

As the concept of Fast motion estimation is very critical in object tracking. Among all search algorithms, the diamond search algorithm is proven for its efficiency in performance with less number of search points. But since a lot of complexity is involved in the searching mechanism, the hardware implementation prevails to be a bottleneck for handheld device applications. The idea behind this work is to reduce the hardware complexity by performing the SAD calculation only in Large Diamond block rather than having in all other blocks. This novelty in the architecture for the Diamond Search Algorithm results in the area reduction of three times and the power reduction of 79%. Hence, this Hardware efficient architecture maintaining the same performance with the reduction in area and power will be useful to applications which require a lesser area.

References

[1]. Chang-Uk Jeong, Takeshi Ikenaga & Satoshi Goto, "An extended small diamond search algorithm for fast block motion estimation," In *Proc. of IEEE International Technical Conference on Circus/Systems, Computers*, (2008).

[2]. Huang, Y.W., Chen, C.Y., Tsai, C.H., Shen, C.F., & Chen, L.G., (2006), "Survey on block matching motion estimation algorithms and architectures with new results", *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, Vol.42, No.3, pp.297-320

[3]. Sumeer Goel, Yasser Ismail, Parimal Devulapalli, Jason McNeely & Magdy A. Bayoumi, "An Efficient Data Reuse Motion Estimation Engine," *In Proc. of IEEE Workshop on Signal Processing Systems Design and Implementation*, (2006).

[4]. Mukherjee, R., Saha, P., Chakrabarti, I., Dutta, P.K. & Ray A.K. (2018), "Fast adaptive motion estimation algorithm and its efficient vlsi system for high definition videos" *Expert Systems with Applications*, Vol.101, pp.159-175.

[5]. Yun Cheng & Min Wu, "A fast motion estimation algorithm based on diamond and triangle search patterns", *In Proc. of IEEE International Conference on Pervasive Computing and Applications*, (2008) pp.419-426.

[6]. Sherief M.Hashimaa, Imbaby I.Mahmoud & Atef A.Elazma "Hardware Implementation of diamond search algorithm for Motion Estimation and Object Tracking", *In Proc. of International Conference on Nuclear and Particle Physics*, (2009).

[7]. Yang Song, Zhenyu Liu, Takeshi Ikenaga, & Satoshi Goto, "Enhanced Strict Multilevel Successive Elimination Algorithm for Fast Motion Estimation," *In Proc. of IEEE International Symposium on Circuits and Systems*, (2007), pp. 3659-3662.

[8]. Shan Zhu & Kai-Kuang Ma (2000), "A new diamond search algorithm for fast block matching motion estimation", *IEEE Transactions on Image Processing*, Vol. 9, No. 2, pp. 287–290.

[9]. Ria & Subhash Chandra Yadav, "A novel diamond search ME algorithm for systolic arrays," *In Proc. of IEEE International Conference on Computing, Communication and Automation*, (2016), pp. 579-581.

[10]. M. Ghanbari (1990), "The cross-search algorithm for motion estimation", *IEEE Transactions on Communications*, Vol. 38, No. 7, pp. 950– 953.

[11]. Gwo Long Li & Mei-Juan Chen, "Fast Motion Estimation Algorithm by Finite-State Side Match for H.264 Video Coding Standard," *In Proc.of IEEE Asia Pacific Conference on Circuits and Systems*, (2006), pp. 414-417.

[12]. Yun Cheng, Xine You, Minlian Xiao & Minlei Xiao, "A Modified Diamond Search algorithm," *In Proc. of IEEE International Symposium on IT in Medicine and Education*, (2011), pp. 481-485.

[13]. Prayline Rajabai C & Sivanantham S, "Review on Architectures of Motion Estimation for Video Coding Standards" *International Journal of Engineering and Technology(UAE)*, 7(4.10 Special Issue 10), pp. 928-934.

[14]. Santosh, C., Rajabai, C. P., & Sivanantham, S., "A SAD architecture for variable block size motion estimation in H.264 video coding," *In Proc. of 2017 International Conference on Microelectronic Devices, Circuits and Systems*, (2017) 2017-January 1-5.