Indonesian Journal of Electrical Engineering and Computer Science
Vol. 23, No. 2, August 2021, pp. 802~810
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v23.i2.pp802-810

Efficient reconfigurable architecture for moving object detection with motion compensation

Sridevi N., M. Meenakshi
Departement of EIE, Dr Ambedkar Institute of Technology, Bangalore, Karnataka, India
VTU, Belagavi, Karnataka, India

Article Info

Article history:

Received Feb 8, 2021
Revised Jun 25, 2021
Accepted Jul 7, 2021

Keywords:

FPGA architecture
Motion compensation
Motion estimation
Video processing

ABSTRACT

The detection and tracking of object in large data surveillance requires a proper motion estimation and compensation techniques which are generally used to detect accurate movement from video stream. In this paper, a novel hardware level architecture involving motion detection, estimation, and compensation is proposed for real-time implementation. The motion vectors are obtained using 16×16 sub-blocks with a novel parallel D flip flop architecture in this work to arrive at an optimised architecture. The sum of absolute difference (SAD) is then calculated by optimized absolute difference and adder blocks designed using kogge-stone adder which helps in improving the speed of the architecture. The controller block is designed by finite state machine model used for synchronization of all the operations. Further, the comparator and compensation blocks are optimized by using basic logical elements and the Kogge-stone adder. Finally, the proposed architecture is implemented on Zynq Z7-10 field-programmable gate array (FPGA) and simulated using System Generator tool for real time traffic signal. The hardware and software parameters are compared with the existing techniques which shows that the proposed architecture is efficient than existing methods of design.

Corresponding Author:

Sridevi N.
Department of Electronics and Instrumentation Engineering
Dr Ambedkar Institute of Technology
Outer Ring Road, Mallathahalli, Bangalore, Karnataka, India
Email: sridevee@gmail.com

1. INTRODUCTION

The moving object detection algorithms are mainly used in different applications such as surveillance and traffic monitoring [1]. Among these applications, movement detection and corrections are mainly used in traffic monitoring systems. For the case of traffic signal, the processing algorithm and architecture must detect and correct the motion at high speed which requires to implement the algorithm and architecture in application-specific integrated circuit (ASIC) level hardware. Detecting the target from static camera is simple and easy than those from the moving camera which involves the estimation and compensation of global motion caused by the camera, which is mounted on moving platform [2]. Moreover, block based motion estimation is one of the most important approach to estimate the motion caused by the movement of the object that are moving in the video stream. Different types of motion estimation algorithms are proposed to estimate the motion from the input video streams which are then used to detect components that are in motion. The accuracy of this estimation is directly related to the overall accurate detection. The three step search (TSS), the four step search (FSS), diamond search, successive elimination search and adaptive search window size [3]-[5] are some of most commonly used motion estimation algorithms, which generates high quality motion estimation, among those algorithms, full search motion estimation is widely used to estimate motion due to its various advantages over other estimation technique.

Moreover, due to the complex data flow of most of those techniques, very large-scale integration (VLSI) level implementations are not possible. Although some VLSI architectures are present to implement motion estimation techniques, the detection accuracy is low and the overall hardware utilizations are high, which makes those architecture are not suitable for real time high speed applications [6]. On the other hand, the full search motion estimation algorithm shows regular data flow which makes it suitable to implement it through VLSI architecture. In the case of motion estimation the current and reference block in adjacent frames are used to define motion vector [7], [8], which is used to estimate required motions. The full search block matching motion estimation is the one most popular motion estimation algorithm. In this case, the current frames are subdivided into a finite number of sub-blocks, which are there used to find best matched blocks in the reference frames. However, Motion based target detection rely on camera motion compensation and correction. Motion estimation and compensation plays an important role in the global motion compensation. Which requires the motion in the video frames to be estimated and the unwanted motion induced due to the movement of the camera to be compensated. Hence, to address the facts discussed above in this paper, an efficient hardware architecture to implement a full search motion estimation algorithm to detect moving objects and compensate for the ego-motion is presented.

The general block diagram of the motion estimation and global motion compensation is given in Figure 1 and the operation is explain below. In the preprocessing the input video stream is converted into number frames which are then converted into different format for hardware processing. The processed frames is then given to the motion estimation block to find the moving parts in the video. The vectors thus detected not only contains the movement due to the target present in the video but also the movement caused by the movement of the sensor that is used to capture the video. Therefore it is required to eliminate the global motion caused by the camera which is achieved in this work with the addition of compensation block.

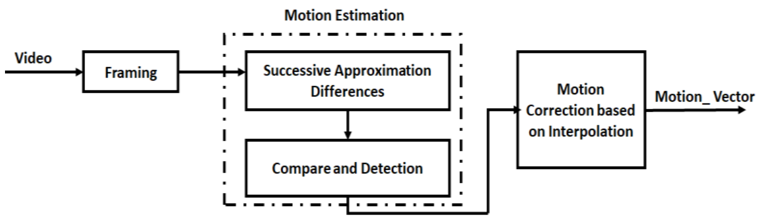
Motion Estimation
Video
Framing
Successive Approximation Differences
Compare and Detection
Motion Correction based on Interpolation
Motion_ Vector

Figure 1. Working method of proposed motion estimation and correction

The architecture proposed here is designed using very high speed integrated circuit (VHSIC) hardware description language (VHDL) language and validate on the FPGA platform. The contribution of this work is highlighted below:

- At first the controller block is optimized by designing it using simple counter.
- The use of kogge-stone adder improves the speed of operation of the arhitecture.
- Further, to reduce the overall hardware utilizations data reuse technique is used.
- The proposed architecture is evaluated by considering three different traffic scenarios.
- Further, the performance of the architecture is evaluated using true and false detection rate.

Organization: The organization of the remaining topics of this paper is: section 2 explains the novel hardware architecture for motion estimation, followed by results and discussion in section 3 and finally conclusion are drawn in section 4.

2. RELATED WORK

Several researchers contributed numerous methods to estimate and compensate the motion, few are discussed here. Pakdaman *et al.* [9] presented a scalable fast motion estimation algorithm through low complex and scalable techniques which uses most popular “test zone (TZ)” for motion estimation algorithm and is useful to get efficient video coding (HEVC). Here in this paper a single reliable starting point is found to replace the first step of the TZ search algorithm. The computational complexity, data dependency with neighboring blocks and deficiency of computational adjustability are the drawbacks of this algorithm. In paper [10] a comparative study of various motion estimation algorithms and operational cycle for motion vector search is evaluated by comparing the existing conventional full search algorithm with the new algorithm developed by them. From the results they concluded that operation cycle of proposed method in [10] is few times smaller than that of full search algorithm but the circuit size is greater than the full search algorithm. Mogus *et al.* [11] discussed the process of motion estimation which generates motion vectors to determine the compensation from prediction frame. Here, to overcome the drawbacks of the algorithm a block matching motion estimation algorithm is also used. In paper [12] the novel architecture for low-latency and high throughput programmable motion estimator is presented. Here motion is estimated by applying full-search and hierarchical search algorithms. Two-step search on gray coded video frames for motion estimation is proposed in [13]. In this method motion is estimated using two steps. Firstly the basic motion vectors are calculated using most significant bits of gray coded bit planes. In the second stage motion vectors are obtained by using an adaptive search pattern.

Gharavi and Mills [14] proposed a novel block-matching motion estimation to find the best match by considering the behavior of individual pels which plays more active role in estimating the motion and results in better performance than mean-absolute-difference. Bhattacharyya *et al.* [15] are worked on block-based motion estimation technique and implemented six-level nested do-loop full-search block-matching motion estimation algorithm. Here the algorithm is implemented in two phases. In the first phase using 25 movie frames without breaking them into macro-blocks and in the second phase the same is implemented after breaking into the respective macro-blocks. Efficient motion estimation algorithm using only diamond search grid to meet the requirement of portable and low power device is presented in the paper [16] using sub-sampling and pixel truncation to reduce the complexity, so they concluded that proposed diamond grid search (DGS) algorithm takes less number of search point and have comparable peak signal to noise ratio (PSNR) and bit-rate as compared to other state of the art motion estimation algorithm. Reference [17] Addressed the problem of area efficiency of carry select adder by avoiding the use of ZFC and multiplexer. Discrete cosine transform-based motion-estimation (DXT-ME) is presented in the paper [18], this algorithm provides the exact displacement of the object of interest, making it suitable for fine-grained tracking. Boonthep *et al.* [19], parallel hardware architecture for computing ME by using scale-invariant-feature-transform (SIFT) is proposed. Here in this paper the authors applied fast fourier transform (FFT) to reduce the complexity in SIFT algorithm also the features that are detected are fully invariant to image scaling and rotation. The proposed architecture will increase the speed of operation due to the use of koggestone adder and the data reuse technique will optimize the overall architecture.

3. PROPOSED METHOD

The hardware architecture proposed in this paper to implement motion estimation is shown in Figure 2. The architecture mainly includes different types of memory unit, absolute difference block, adder array bock, comparator, and controller unit respectively the 16×16 block size is used as current block to get the motion vectors which are stored in the external memory blocks, which is capable of storing entire frame until it is processed. The current block and the subdivided computational block obtained by dividing the frame into number of sub-blocks are routed through the demux block to distribute them in to three memory blocks namely SUBM1, SUBM2 and SUBM3 respectively. These memories store current sub matrix and adjacent sub matrix pixel values. The stored pixel values are then used to calculate the successive approximation differences (SAD) through which the motions are detected and corrected through compare and correction block. The motion features stored in the motion vector memory are then used as references to the next frame. The controller block is used to control the overall operation by controlling the data-path of the architecture.

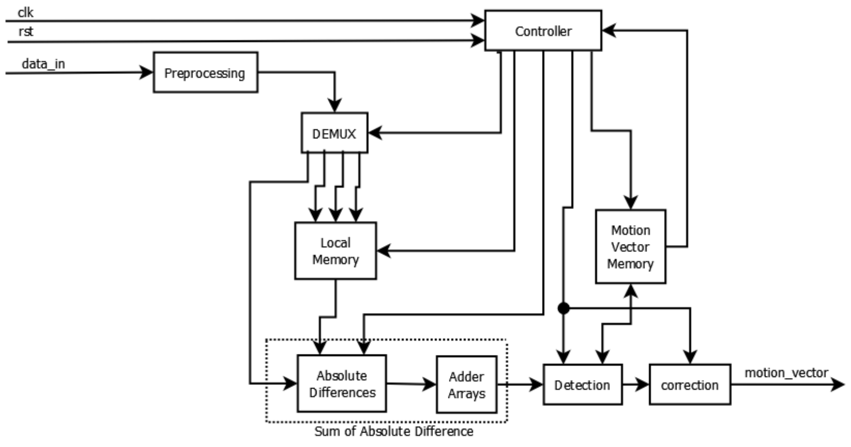

Figure 2. Hardware architecture of motion estimation and correction

3.1. Preprocessing

The direct video frames are not compatible and portable for hardware processing due to different formats of videos existing. As a result, it is necessary to convert it into a number of frames which are performed by system generator tool with the help of MATLAB tool. It is normally used to convert the input video into a finite number of frames of standard size.

3.2. Controller

The controller block is used to control the overall data flow either by activating or deactivating the required blocks. In the beginning of the operation, the select line of the DEMUX is set to 0 which allows the first sub-matrix pixels to enter in sub-memory 1. After 16 clock cycles, the absolute difference block starts calculating the absolute difference values from the current block. At this time, the controller block selects the sub-memory 2 through DEMUX block. The absolute difference of two sub-matrixes starts calculating values till 49th clock cycle and the final SAD values starts at 50th clock cycle and it is implemented by simple counter logics.

The hardware architecture to design the controller block is given in Figure 3 which consists of counter, decision maker block and encoder block. The counter block starts counting at every rising edge of the clock pulse when reset (rst) signal is high which is then used by decision maker block to decide the correct sequence of sub-blocks to be activated and then this is encoded by the encoder block to activate the processing elements in correct sequence. After the motion vector calculation of three consecutive 16×16 sub-matrices are completed, the entire controller is reset to its initial condition and the operation starts from the beginning.

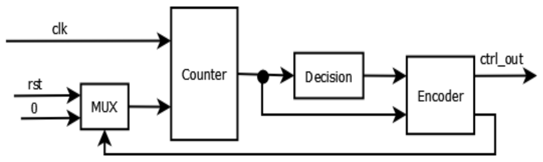

Figure 3. Proposed controller architecture

3.3. Motion detection

The motion vectors are used to detect the movement from any video sequence using absolute difference, array of adders and decision maker block.

3.3.1. Absolute difference

To calculate the absolute difference between neighboring frames, it is essential to store the pixel of these frames into temporary memory for processing and the architecture is given in Figure 4.

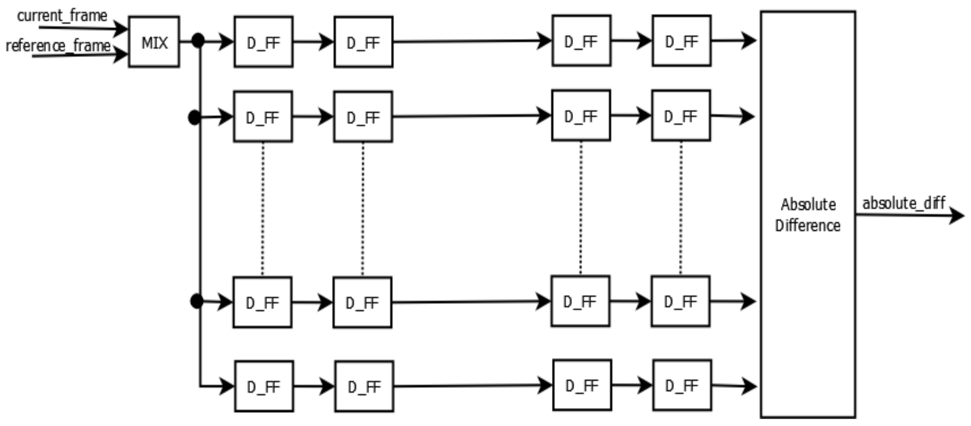

Figure 4. Temporary storage of frames

The shift register architecture designed using D flip-flops are mainly used for this purpose. The architecture which is used to store the frames are given in Figure 3 where the D_FF block is used to store the pixel values which are then used to calculate the absolute difference. The parallel architecture of the D_FF is an advantage for fast processing of the data compared to existing techniques.

The absolute difference values between the current block and adjacent blocks are calculated in this block. For motion estimation purpose, it is essential to perform the absolute difference (AD) at block level which requires a high speed subtractor. As a result, the normal subtractor architecture is replaced by the kogge-stone adder [20] in binary arithmetic. The equation for absolute difference is given below in (1):

$$Absolute\ Difference\ (AD) = |A - B| \quad (1)$$

Where, A and B are the pixels of current and reference frame respectively.

The hardware architecture is used to obtain the absolute difference is shown in the Figure 5 where the use of kogge-stone adder resulted in optimization of the entire architecture. The subtraction is modeled by addition in binary arithmetic which is then used to find the sign of the resultant data through concatenation block. Depending upon the output of the concatenation block, the MUX block send the resultant value or 2's complement of the resultant value to the output which is the absolute difference value of both input signal. For proper optimization, the 2's complement is implemented by kogge-stone adder block only.

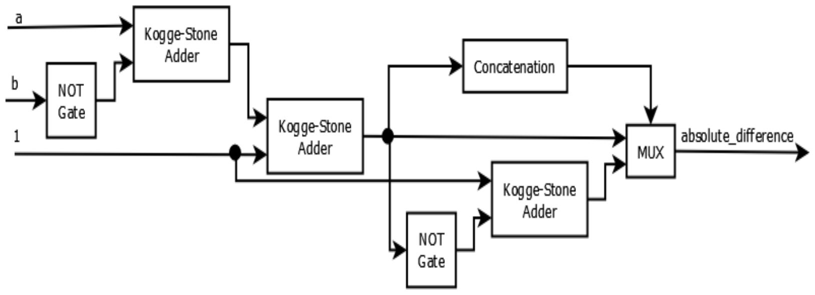

Figure 5. Hardware architecture of absolute difference

3.3.2. Adder array

The Kogge-Stone adder [20] architecture is used to build the adder array in parallel fashion.

3.4. Decision block

The basic comparator block is used to compare the matched frame from stored vectors in the motion vector memory block for accurate motion vector. It is normally done through logical comparator block and the hardware architecture of motion estimation is given in Figure 6.

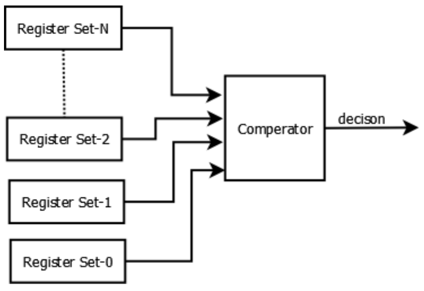

Figure 6. Hardware architecture of motion detection

3.5. Local memory

The pixels present in search area matrix are stored in the local memory unit which consists of three memory blocks namely SUBM1, SUBM2 and SUBM3 respectively. These memories are used to store the three consecutive 16×16 blocks through DEMUX block through select line which is controlled by counter block. The data enters as pixels of 16 bits in row by row from top to bottom direction through counter logic.

3.6. Motion correction

The motion vector of all overlapping blocks present in the corresponding frames must be stored in a memory block for further processing and its architecture is given in Figure 7. The motion vector memory block is used to store the calculated motion vectors from the SAD data through compare and compensate block. It is implemented by block RAM available on the FPGA board. The size of the RAM block depends upon the frame size and the SAD matrix size. In many cases the use of simple comparison technique introduces some false detection which has to be removed by correction block. The N×N blocks are used to interpolate 2N×2N overlapped blocks.

The modules for estimating forward motion, bidirectional motion and smoothing spatial motion are the same as those in the original algorithm [21] with modification of motion compensation architecture with counter controlled registers instead of RAM controlled. The novel architecture used to implement the motion compensation through interpolation [21] is given in Figure 8. The motion vectors obtained are then stored into another temporary memory which are then used for compensating the motion through interpolation array to generate the corrected motion vector. The entire architecture is controlled by the controller block for proper synchronization.

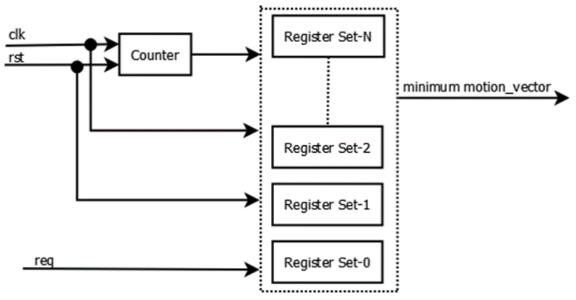

Figure 7. Hardware architecture of motion vector memory

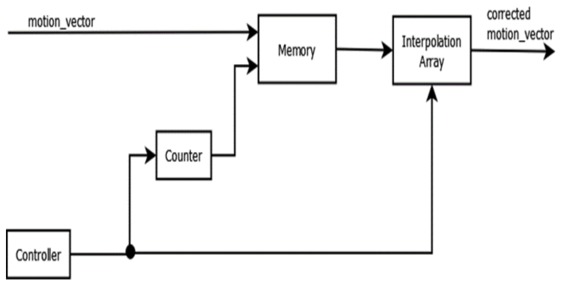

Figure 8. Hardware architecture of motion correction

4. RESULTS AND DISCUSSION

The motion estimation and correction architecture are designed using VHDL and validated the design using Xilinx EDA tool for Zynq Z7-10 platform.

4.1. FPGA implementation

The architecture proposed in this work is implemented on digital Zynq Z7-10 FPGA. The standard VHDL coding technique is used to design the architecture and is simulated using Xilinx 14.5 design suite tool. The hardware utilization of the proposed motion estimation and compensation architecture is given in the Table 1 with its intermediate components.

Table 1. Hardware utilizations of motion estimation and correction architecture

Parameters	SAD	Comparator	Control	Memory	Total Module
FPGA	Zynq Z7-10 (xc7z010-1clg400c)				
Slice registers	6144	135	21	6154	12450
Slice LUTs	6299	146	460	2184	908
LUT-FF pairs	2910	86	12	1850	517
Maximum frequency (MHz)	1432.665	321.404	650.347	707.214	321.404

The System generator tool is used to input the real time video in the designed architecture, through standard interfacing methods. Here the input video is fed to the FPGA using image processing toolbox in MATLAB. The motion vectors thus calculated on FPGA are then used by the MATLAB tool to insert boxes for better visualization.

4.2. Performance analysis

The performance of the proposed architecture in terms of hardware utilizations and detection accuracy are discussed as follows.

4.2.1. Simulation

The designed architecture is simulated by taking three different scenarios low density traffic, high density traffic and video steam captured using moving camera to evaluate the detection accuracy of the system. The statistical parameters such as detection accuracy, true detection rate, false detection rate and not detected rate [22] are calculated to validate the designed architecture.

Figures 9-11 shows one random frame of the entire simulation for normal traffic, dense traffic and video taken from moving camera respectively. From the calculation it is found that the true detection rate is 93.79% and false detection rate of 4.21%. Under dense traffic condition the true detection rate of 93.01% and the false detection rate is 13.02% is obtained by randomly selecting the frame from the entire simulation.

However**,** the video taken from moving camera shows the true detection rate of 92.93% and false detection rate of 11.91 %. The randam frame taken from the entire simulation result. Evaluation of various parameters are given in Table 2. Higher value of true detection rate indicates the better detection accuracy. Low value of not detected rate indicates correct detection. Through the analysis it is observed that the developed algorithm is capable of detecting the objects that are in motion.

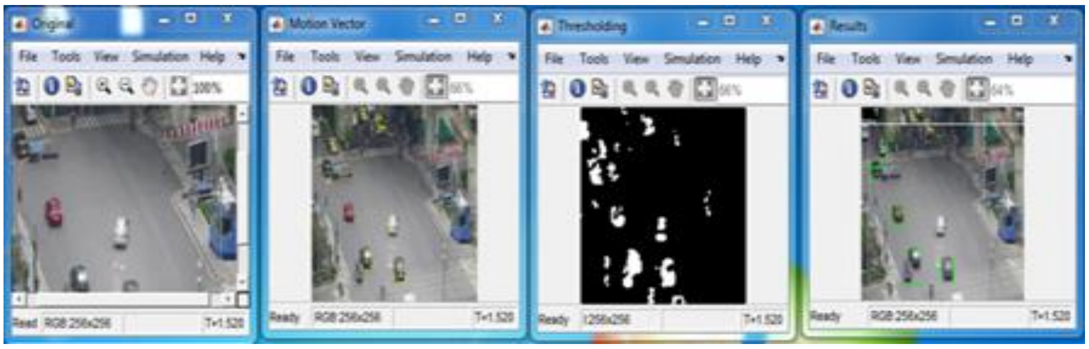

Figure 9. Simulation results of normal traffic flow

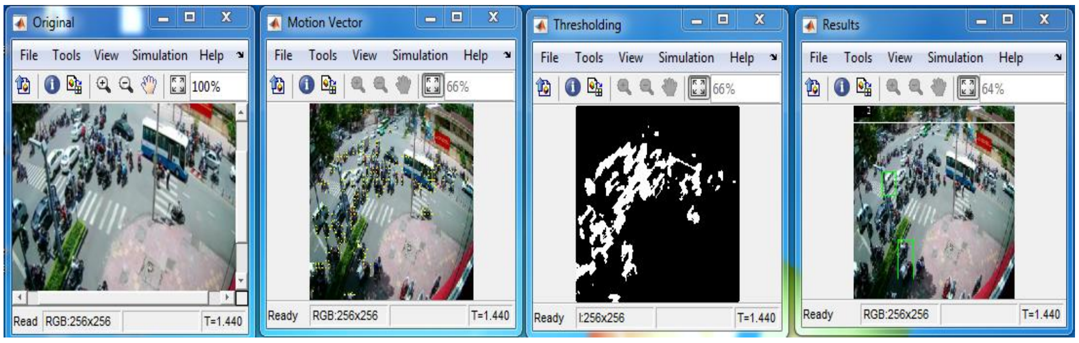

Figure 10. Simulation results of normal dense flow

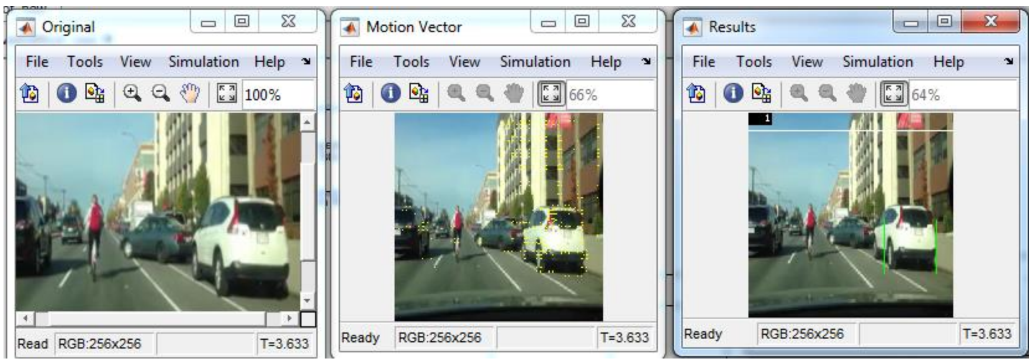

Figure 11. Simulation results of detection when the camera is in motion

Table 2. The performance parameters with respect to normal traffic scenario

Different Senario	True Detection Rate	False Detection	Not Detected Rate
Normal	93.79%	4.21 %	3.11%
Dense	93.01%	13.02 %	6.02%
Moving Camera	92.93%	11.91 %	9.55%

4.2.2. Comparisons with existing techniques

The detection accuracy of the proposed architecture is compared with the existing architectureto check the accuracy which is given in Table 3. From the table it can be seen that the proposed architecture is able to detect the moving object accurately than existing.

Table 3. Detection accuracy comparisons of proposed architecture with existing

Authors	Techniques	Maximum Accuracy
Chih-Yang *et al.* [23]	Image Bit-Planes for Real-Time Video Surveillance	86.30%
Yu and Fenfen [24]	Optical flow	89.00%
Sridevi *et al.* [25]	Gaussian Mixture Model with Morphological filter	93.18%
Haidi Zhu *et al.* [26]	YOLO	88.59%
Proposed Architecture	SAD Based Comparison	93.79%

The hardware utilizations of the proposed architecture are compared with the proposed motion estimation and compensation architecture presented by Jingyan and Peng [27] which is given in the Table 4. The architecture presented by Jingyan and Peng [27] was implemented on Zynq FPGA with high level programming method (C/C++ language) which are not useful to design parallel optimized hardware architecture. On the other hand, the proposed architecture is implemented using HDL language with architectural modification to get optimized hardware utilizations

Table 4. Hardware comparisons of proposed architecture with existing

Parameters	Jingyan and Peng [27]	Jaechan Cho *et al.* [28]	Proposed architecture
FPGA	Zynq-7	Virtex-5	Zynq-7
Slice Register	12767	13245	12450
Slice LUTs	37761	----	908
LUT-FF Pairs	27875	----	517
Maximum Frequency (MHz)	128	200	321.404

5. CONCLUSION

Motion estimation and compensation is one of the most important method to detect and correct the global motion caused by the camera. Further, many computer vision applications demand high speed, less hardware utilization in the architecture. In this paper, considering the full search block based method an efficient hardware architecture for motion estimation and compensation for proper object detection is developed. Further, the architecture is evaluated by considering three different traffic conditions. To optimize the hardware utilization, the sum of absolute difference architecture is optimized by using kogge-stone adder. The counter logic is used to optimize the controller architecture by adopting finite state machine (FSM) modeling there by 12450 slice register and 517 LUT-FF pairs are used to develop the architecture. The entire architecture is implemented on Zynq-Z7-10 FPGA. The experimental validation proved that the accuracy of 93.79% and the operating speed of 321.404 which is high compared with some of the existing techniques.

REFERENCES

[1] Zhu, J., Wang, Z., Wang, S., and Chen, S., “Moving Object Detection Based on Background Compensation and Deep Learning,” *Symmetry*, vol. 12, no. 12, pp. 1-12, 2020, doi: 10.3390/sym12121965.

[2] Bhattacharya, S., Idrees, H., Saleemi, I., Ali, S., and Shah, M., “Moving Object Detection and Tracking in Forward Looking Infra-Red Aerial Imagery,” *Machine Vision Beyond Visible Spectrum*, pp. 221-252, 2011, doi: 10.1007/978-3-642-11568-4_10.

[3] Basha, S., and Kannan, M., “Literature survey on motion estimation techniques,” *International Journal of Engineering & Technology*, vol. 7, no. 2.12, p. 394, Apr. 2018, doi: 10.14419/ijet.v7i2.12.11358.

[4] Verma, N., Sahu, T., and Sahu, P., “Efficient motion estimation by fast three step search algorithms,” *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, vol. 1, no. 5, pp. 380-385, Nov. 2012.

[5] Ahmadi A. and Azadfar M. M., “Implementation of Fast Motion Estimation Algorithms and Comparison with Full Search Method in H.264, ” *IJCSNS International Journal of Computer Science and Network Security*, vol. 8, no. 3, pp. 139-143, Mar. 2008.

[6] Sanz C., Garrido M. J., and Meneses J. M., “VLSI Architecture for Motion Estimation using the Block-Matching Algorithm,” *Proceedings of the European conference on Design and Test*, 1996, pp. 310-314, doi: 10.1109/EDTC.1996.494318.

[7] Yousaf, A., Hanif, M. S., Khan, M. J., Iqbal, M., and Khurshid, K., “Robust and computationally efficient online image stabilisation framework based on adaptive dual motion vector integration,” *IET Computer Vision*, vol. 13, no. 5, pp. 461-468, 2019, doi: 10.1049/iet-cvi.2018.5368.

[8] Tang, J. W., Shaikh-Husin, N., Sheikh, U. U., and Marsono, M. N., “FPGA-Based Real-Time Moving Target Detection System for Unmanned Aerial Vehicle Application,” *International Journal of Reconfigurable Computing*, vol. 2016, no. 3, pp. 1-16, 2016, doi: 10.1155/2016/8457908.

[9] Pakdaman, F., Hashemi, M. R., and Ghanbari, M., “A low complexity and computationally scalable fast motion estimation algorithm for HEVC,” *Multimedia Tools and Applications*, vol. 79, pp. 11639-11666, 2020, doi: 10.1007/s11042-019-08593-y.

[10] Bnadou, R., Hiramori, M., Iwade, S., Makino, H., Yoshimura, T., and Matsuda, Y., “A Study on Motion Estimation Algorithm for Moving Pictures,” *2016 IEEE 5th Global Conference on Consumer Electronics*, pp. 1-3, 2016, doi: 10.1109/GCCE.2016.7800439.

[11] Mogus, F. A., Liu, X., and Wang, L., “Evaluation of the Performance of Motion Estimation Algorithms in Video Coding,” In *The 2nd IEEE International Conference on Information Science and Engineering*, 2010, pp. 3693-3696, doi: 10.1109/ICISE.2010.5691643.

[12] Shu, Q., and Chen, H., “An efficient implementation of motion estimation algorithms,” *Proceedings of 4th International Conference on Solid-State and IC Technology*, 1995, doi: 10.1109/ICSICT.1995.503394.

[13] Chatterjee, S. K., and Vittapu, S. K., “An Efficient Motion Estimation Algorithm for Mobile Video Applications,” *2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP)*, 2019, pp. 1-5, doi: 10.1109/ICACCP.2019.8882948.

[14] Gharavi H. and Mills M., “Blockmatching Motion Estimation Algorithms,” *IEEE Transactions on Circuits and Systems*, vol. 37, no. 5, pp. 649-651, 1990, doi: 10.1109/31.55010.

[15] Bhattacharyya, D., Chakrabarti, A., and Misra, S.,“Design and Simulation of Parallel Algorithms for Motion Estimation,” *10th IEEE International Conference on Information Technology*, 2007, pp. 29-34, doi: 10.1109/ICIT.2007.44.

[16] Singh K., and Ahamed, S. R., “A New Motion Estimation Algorithm for High Efficient Video Coding Standard,” *2015 Annual IEEE India Conference (INDICON)*, 2015, pp. 1-5, doi: 10.1109/INDICON.2015.7443614.

[17] Tapasvi, B., Sinduri, K. B., Lakshmi, B. G. S. S. B., and Kumar, N. U., “Implementation of 64-Bit Kogge Stone Carry Select Adder with ZFC for Efficient Area,” In *2015 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, 2015, pp. 1-6, doi: 10.1109/ICECCT.2015.7226154.

[18] Mishra P., Apoorva R., Parvatikar B. B., and Nair L., “Architectures for FPGA-Based Implementation of Motion Estimation of Dynamic Obstacles for Autonomous Robot Navigation,” *2011 Third International Conference on Computational Intelligence, Communication Systems and Networks*, 2011, doi: 10.1109/CICSyN.2011.68.

[19] Boonthep N., Chamnongthai K., and Phensadsaeng P., “A FPGA-based SIFT Architecture for Motion Estimation in Video Coding,” *2018 Global Wireless Summit (GWS)*, 2019, pp. 383-388, doi: 10.1109/GWS.2018.8686403.

[20] Xiang L. M., Zabidi M. A., M., Awab, A. H., and Ab Rahman, A. A. H., “VLSI Implmentation of a Fast Kogge-Stone Parallel-Prefix Adder,” *Journal of Physics: Conference Series*, 2018, vol. 1049, no. 1, pp. 1-10, doi: 10.1088/1742-6596/1049/1/012077.

[21] Zhang D. Y., Ji Y. T., and Wang X. M., “A Weighted Motion Compensation Interpolation Method for Improving Side Information in Distributed Video Coding,” *Applied Mechanics and Materials*, vol. 519, pp. 672-676, 2014, doi: 10.4028/www.scientific.net/AMM.519-520.670.

[22] Sridevi N. and Meenakshi M., “Efficient Moving Vehicle Detection Algorithm for Various Traffic Conditions,” *International Journal of Recent Technology and Engineering (IJRTE)*, pp-6069-6076, Sep. 2019, doi: 10.35940/ijrte.C5619.098319.

[23] Chih-Yang L., Zhi-Yao J., and Wei-Yang L., “Image Bit-Planes Representation for Moving Object Detection in Real-Time Video Surveillance,” *2016 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, 2016, pp.1-2, doi: 10.1109/ICCE-TW.2016.7520949.

[24] Zhang Y. and Wang F., “Improved optical flow algorithm of moving object Detection,” *2015 Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC)*, 2016, pp.196-199, doi: 10.1109/IMCCC.2015.48.

[25] Sridevi N., and Meenakshi M., “Efficient Movement Compensation and Detection Algorithm using Blob Detection and Modified Kalman Filter,” *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, 2020, pp-264-268, doi: 10.1109/ICCES48766.2020.9138031.

[26] Zhu H., Wei H., Li B., Yuan X., and Kehtarnavaz N., “Real-Time Moving Object Detection in High-Resolution Video Sensing,” *Sensors*, vol. 20, no. 12, pp. 1-15, 2020, doi: 10.3390/s20123591.

[27] She J. and Du P., “FPGA-Based Motion Estimation Algorithm Optimization,” *Microprocessors and Microsystems*, vol. 80, pp. 1-13, 2021, doi: 10.1016/j.micpro.2020.103555.

[28] Cho J., Jung Y., Kim D., Lee S., and Jung Y., “Moving Object Detection Based on Optical Flow Estimation and a Gaussian Mixture Model for Advanced Driver Assistance Systems,” *Sensors*, vol. 19, no. 4, pp. 1-14, 2019, doi: 10.3390/s19143217.