iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 5 **Issue:** IV **Month of publication:** April 2017

DOI: http://doi.org/10.22214/ijraset.2017.4243

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

Volume 5 Issue IV, April 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

FPGA Accelerated Pedestrian Detection System

Reema Kalshaonkar[1], Sonia Kuwelkar[2]
[1.2]*Department of Electronics and Telecommunication, GOA University*

Abstract*: *This paper focuses on developing a Pedestrian Detection System that is accelerated using a Zedboard. A part of the design is implemented on FPGA and the remaining is implemented in Matlab R2015a. Histogram of Oriented Gradients (HOG) has been used for detection purpose and Linear Support Vector Machine (LSVM) has been used for classification. This system gave an accuracy of 93.27%, a True Positive Rate of 92.27% and a False Positive Rate of 4%. Also the system was faster than the software implementation.

Keywords*: *Pedestrian detection, HOG, LSVM, feature extraction, classification

I. INTRODUCTION

Pedestrian detection is an important aspect due to the ever-growing number of vehicle to pedestrian accidents every year. Apart from preventing road accidents, it may also be used in security systems, robotics industry etc. However the system was mainly developed to be used in Advanced Driver Assistance System (ADAS). As per World Health Organisation (WHO) 22% of 1.25 million people die in road accidents caused by vehicle to pedestrians. Hence it's the need of hour to prevent such a misery.

II. RELATED WORK

A lot of research work has been done with respect to pedestrian detection detection system and is still going on. The main requirements of the system are accuracy and speed. The system comprises of feature extraction of an image, followed by classification. The researchers have tried to use the best feature extraction algorithm and the best classification algorithm to meet the needs of real time system. In 1995, there was a major success in this field when Dalal and Triggs came up with a system using HOG and LSVM in [1]. The HOG algorithm itself comprises of several steps that can be modified in a variety of ways. The researchers then experimented with different combinations and came up with their research work trading off between accuracy and speed. In [2], HOG was implemented on zynq with a detection accuracy of 90.2% and FPR of 4%. It used region of Interest (ROI), binaristaion and approximated L1-normalisation that lowered the accuracy. In [3], HOG-DOT algorithm and SVM classification was implemented on FPGA and the accuracy was further improved by CPU filtering. It achieved TPR of 94%. However it's FPR was 5.5 % which was a little high. In [4], HOG-DWT and SVM algorithm was implemented with speed up of 27.12%, but the accuracy was greatly reduced to 85.12%. The algorithm proposed in [5] used sobel filter and L2-normalisation for high accuracy. However it wasn't implemented on any standard dataset. In [6], [7], [8] HOG-LBP algorithm was implemented for higher accuracy but the increased computation reduced the speed greatly. In [9], Gaussian filter was used with HOG for better accuracy and virtex-5 FPGA implementation for higher speed. In [10], normalisation was replaced with binarisation to reduce the latency and complexity of the algorithm. However this reduced the overall accuracy. This was further implemented on low end Spartan-3e FPGA for higher speed. In [11], bigger block size and L1-normalisation further increased the speed but greatly decreased the accuracy. Thus we see that HOG algorithm gives the best accuracy with high speed required for real time application. Also SVM is the best classification technique for further getting higher speed. It is based on a well-defined mathematical concept unlike neural networks. Also SVM has less computation than Adaboost classification algorithm. In [12] it has been shown why SVM performs the best with HOG features. Since these algorithms involve large parallel computation it can be implemented on parallel architecture like FPGA or GPU. GPU consumes more power than FPGA and hence FPGA is preferred to GPU for further hardware acceleration in this paper.

III. ALGORITHM FOR DETECTION AND CLASSIFICATION

The pedestrian detection system takes an input image followed by a pre-processing step. This step scales the image to 128x64 size and converts the image to a gray image. The next step includes the feature extraction step which uses the Histogram of Oriented Gradients (HOG) algorithm that is further explained in the paper. The classification and pre-defined model uses the concept of Linear Support Vector Machine (LSVM). The block diagram for pedestrian detection system is shown in Fig. 1.

Volume 5 Issue IV, April 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

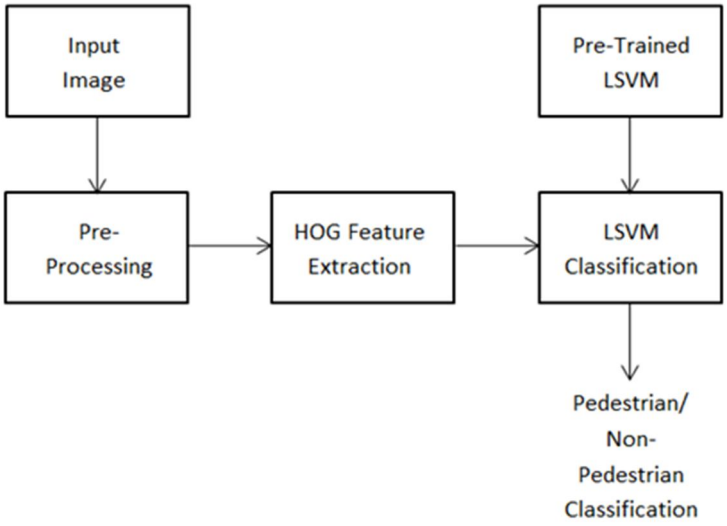


Fig. 1 Pedestrian Detection System

A. Histogram of Oriented Gradients (HOG)

HOG is used for feature extraction. For an image of 128x64 it gives an array of 3780 features. The steps involved in HOG is shown in Fig. 2 and further described in detail.

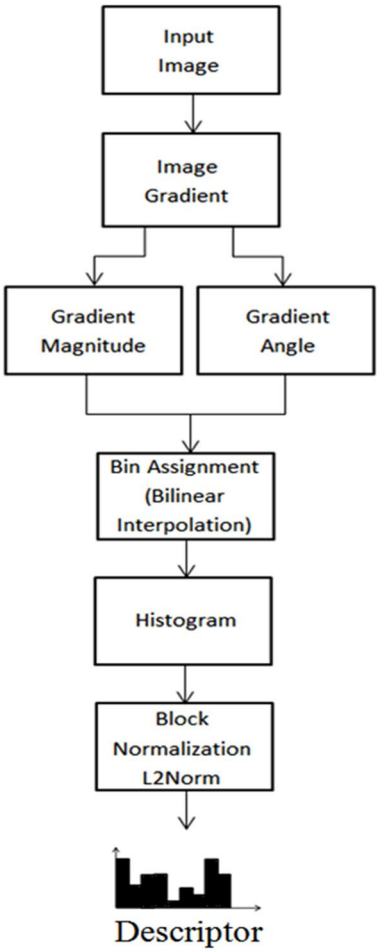


Fig. 2 Block diagram of HOG

Volume 5 Issue IV, April 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

B. The Steps involved in HOG Algorithm are

1) Image Gradient: The input gray image is convolved with the filter masks for Gradient images in x and y direction. Masks, Mx=[-1 0 1] and My =[-1 0 1]T are used for horizontal and vertical direction respectively. Using (1) we compute gradient image in x direction, Fx and gradient image in y direction, Fy of the raw image, I.

$$Fx = Mx * I \qquad (2)$$
$$Fy = My * I$$

2) Gradient Magnitude: The gradient magnitude is computed using (3).

$$|F(x,y)| = \sqrt{Fx^2 * Fy^2} \qquad (3)$$

3) Gradient Angle: The gradient angle is computed using (4).

$$\theta = \arctan\frac{Fy}{Fx} \qquad (4)$$

4) Cell Histogram: The image is divided into equal regions of 8x8 cells as shown in Fig. 3. Cell histogram is then computed for each cell. Histogram is an array of 9 bins in our case. We have used orientation binning to form a histogram of cell as shown in Fig.4. We have used unsigned angles and the angles between 0° to 180° are divided in equal sections of 20° each. Every span of 20° then corresponds to its respective histogram bin i.e. all angles in between 0° to 20° correspond to bin 1, angles between 20° to 40° correspond to bin 2 and so on shown in Fig. 5. For each pixel in a cell, we add its magnitude to the bin that corresponds to its angle. However the magnitude is not added directly to the bins, instead it's bi-linearly interpolated between the bin it belongs to and the next higher bin. Hence, we get a cell histogram as shown in Fig. 6.

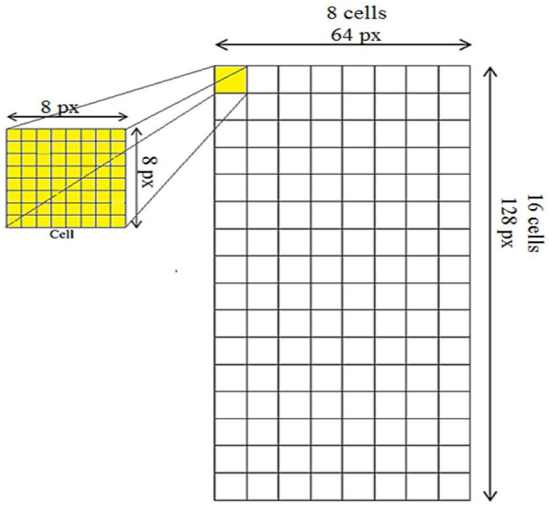


Fig. 3 Division into cells

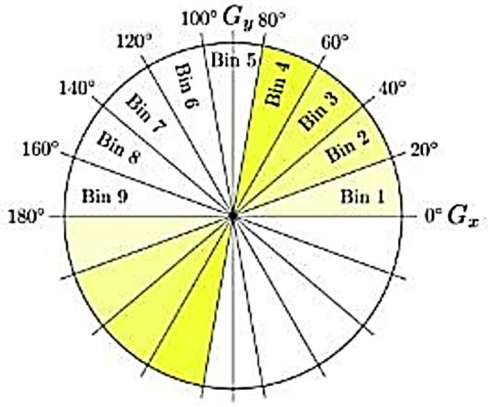


Fig. 4 Division of angles into bins

Volume 5 Issue IV, April 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

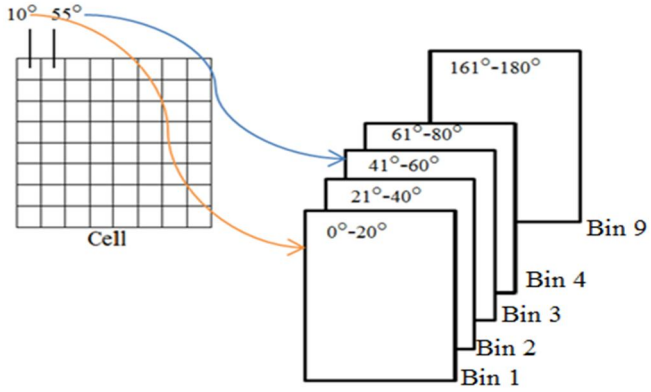


Fig. 5 Orientation Binning

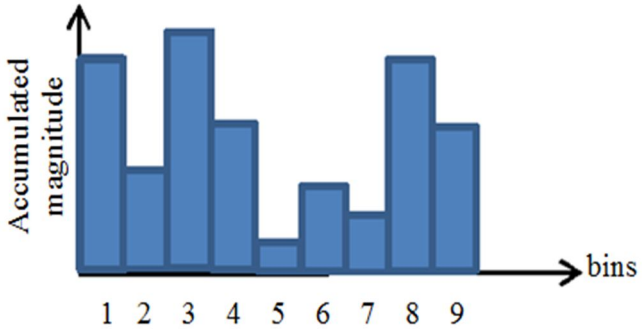


Fig. 6 Cell Histogram

5) Block Histogram: The image is divided into blocks of 2x2 cells with 50% overlap as shown in Fig. 7. We simply concatenate the cell histograms of each block to get a block descriptor of 36 elements as shown in Fig. 8. For better detection accuracy the block descriptor is further normalised using L2-normalisation technique as in (5). In (5) v_k is the normalised block histogram, $\|v_k\|$ is absolute normal of k^{th} block and is a small constant to prevent the result from being infinite.

$$v = \frac{v_k}{\sqrt{\|v_k\|^2 + \varepsilon}} \quad (5)$$

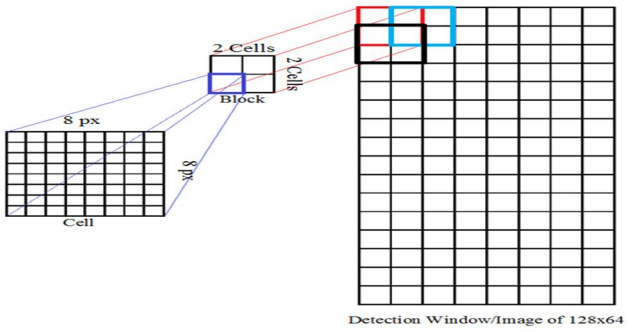


Fig. 7 Image division into blocks

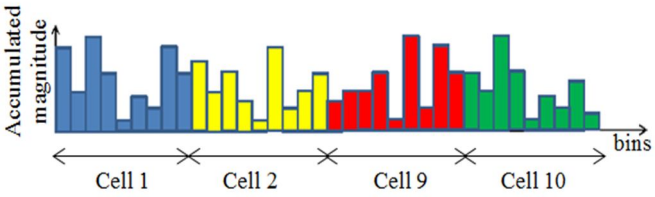


Fig. 8 Block Histogram

Volume 5 Issue IV, April 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

6) Image Histogram: An image of 128x64 has 105 blocks. Hence a histogram of image is computed by concatenating block feature vectors resulting in an image descriptor of 3780 features. The HOG descriptor of an image is depicted in Fig. 9. The descriptor is a representation of edge and orientations of an image.

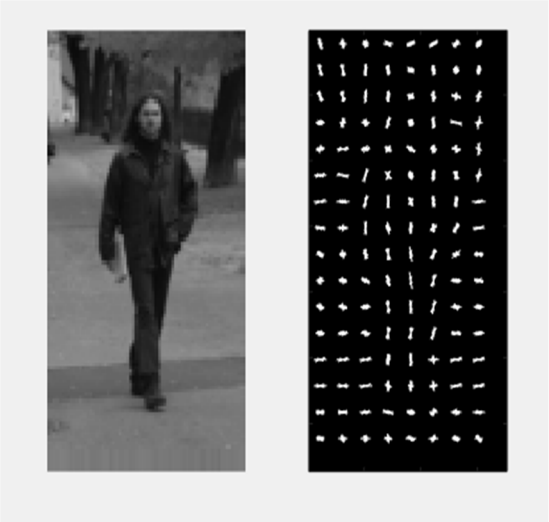


Fig. 9 HOG descriptor of an image

C. Linear Support Vector Machine (LSVM)

SVMs are widely used today because of its speed and simplicity. It aims to find the best separating hyperplane between the two classes of classification in our system. The concept of LSVM is depicted in Fig. 9. The largest margin between two classes is the hyperplane. The largest distance between the two classes is the margin and support vectors are the feature vectors closest to the hyperplane.

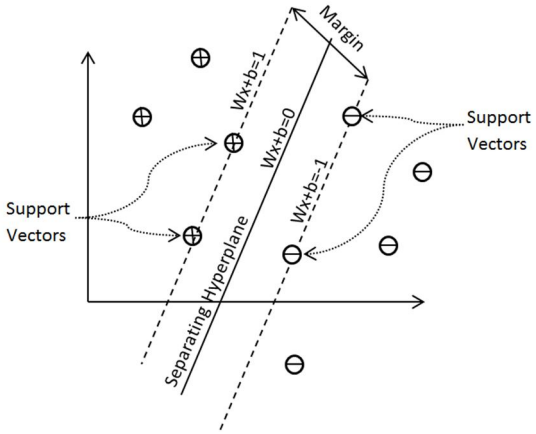


Fig. 10 LSVM Model

A hyperplane is uniquely defined by a normal vector or weight vector, w and an offset or bias, b. The LSVM algorithm is used further to find the weight vector and bias from the feature vectors of the training images. This results in a pre-trained SVM model. The steps of LSVM algorithm are detailed below:

1) Augment the Support Vectors: The support vectors or the feature set of every training image is augmented with 1 which is considered as a bias input.

2) Determine Alpha Constants, αi: For a training set of three datasets and hence three support vectors the LSVM architecture is shown in Fig. 11. The constants are then found using (6). The equations representing pedestrian class were equated to +1 and others were equated to -1.

$$\alpha_1\Phi(s_1).\Phi(s_1) + \alpha_2\Phi(s_2).\Phi(s_1) + \alpha_3\Phi(s_3).\Phi(s_1) = +1$$
$$\alpha_1\Phi(s_1).\Phi(s_2) + \alpha_2\Phi(s_2).\Phi(s_2) + \alpha_3\Phi(s_3).\Phi(s_2) = -1 \qquad (6)$$

Volume 5 Issue IV, April 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

$$\alpha_1\Phi(s_1).\Phi(s_3) + \alpha_2\Phi(s_2).\Phi(s_3) + \alpha_3\Phi(s_3).\Phi(s_3) = -1$$

Φ is a mapping function which is an identity function for a LSVM And hence we get (7). The tilde symbol is used for support vectors to indicate that it is augmented with 1.

$$\begin{aligned} \alpha_1\tilde{s_1}.\tilde{s_1} + \alpha_2\tilde{s_2}.\tilde{s_1} + \alpha_3\tilde{s_3}.\tilde{s_1} &= +1 \\ \alpha_1\tilde{s_1}.\tilde{s_2} + \alpha_2\tilde{s_2}.\tilde{s_2} + \alpha_3\tilde{s_3}.\tilde{s_2} &= -1 \\ \alpha_1\tilde{s_1}.\tilde{s_3} + \alpha_2\tilde{s_2}.\tilde{s_3} + \alpha_3\tilde{s_3}.\tilde{s_3} &= -1 \end{aligned} \quad (7)$$

3) Weight Vector and Bias, b: We use (8) to find the augmented weight vector, $\tilde{w}$. The last element of this vector is the bias, b and the remaining elements form the weight vector, w.

$$\tilde{w} = \sum_i \alpha_i\tilde{s_i} = [\text{w b}] \quad (8)$$

4) Classification: The weight vector, w and bias, b represent the predefined model and used to classify testing image into pedestrian and non-pedestrian classes. This is done using (9) and if the result is positive it is a pedestrian class else a non-pedestrian class.

$$f(x) = sgn(w^T \times x + b) \quad (9)$$

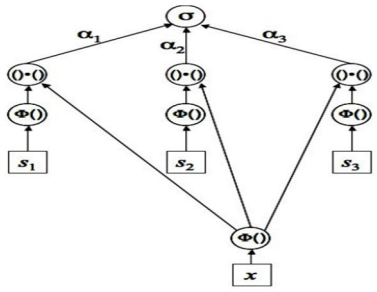


Fig. 11 LSVM Architecture

IV. SOFTWARE SIMULATION-MATLAB

The design was coded and simulated in Matlab R2015a. The INRIA dataset was used for simulation as it is one of the most widely used standard dataset. The details of the INRIA dataset in [16] are shown in Table I.

TABLE I INRIA DATASET

Image Set	Total No.
Training Images	3327
Positive training Images	2416
Negative Training Images	911
Test Images	1426
Positive Test Images	1126
Negative Test Images	300
Training Images	3327

During the training phase, all the training images feature vectors were extracted and used to form the LSVM model. [13], [14], [15] were used in designing the LSVM. This model hence consists of a weight vector, w and bias, b. During the testing phase, the input image is processed to get the feature vector. This feature vector was then used to classify the image into one of the two classes using the trained LSVM model. The results obtained for the INRIA dataset are summarised in Table II. The execution time was found to be 0.56 seconds per image.

Volume 5 Issue IV, April 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

TABLE III SIMULATION RESULT

Parameter	Value
Correctly Detected Images	1327
Detection Accuracy	93.05%
False Negative Images	87
True Positive Images	1039
TPR	92.27%
False Positive Images	12
Correctly Detected Images	1327

V. HARDWARE ACCELEARATION USING FPGA

For a real time application we need a system much faster than the software designed system. This can be achieved using hardware accelerated system. For this purpose we use Xilinx Zynq-7000 EPP ZC702 Evaluation Kit or Zedboard. Some steps of the HOG algorithm are made to run on FPGA to increase the overall speed of the system. These include the gradient computation, gradient magnitude and gradient angle step. A Simulink model as shown in Fig. 12 is designed with an atomic subsystem marked in blue. This subsystem is converted into IP core using the HDL workflow advisor. The generated IP core is shown in Fig. 13. The subsystem is further synthesised and implemented using Vivado 14.2 and the generated bitstream is used to program the FPGA. The rest of the design is implemented on Matlab itself. Thus dividing the workload between the FPGA and Matlab reduces the overall execution time to a great extent while retaining the accuracy of a software implementation.

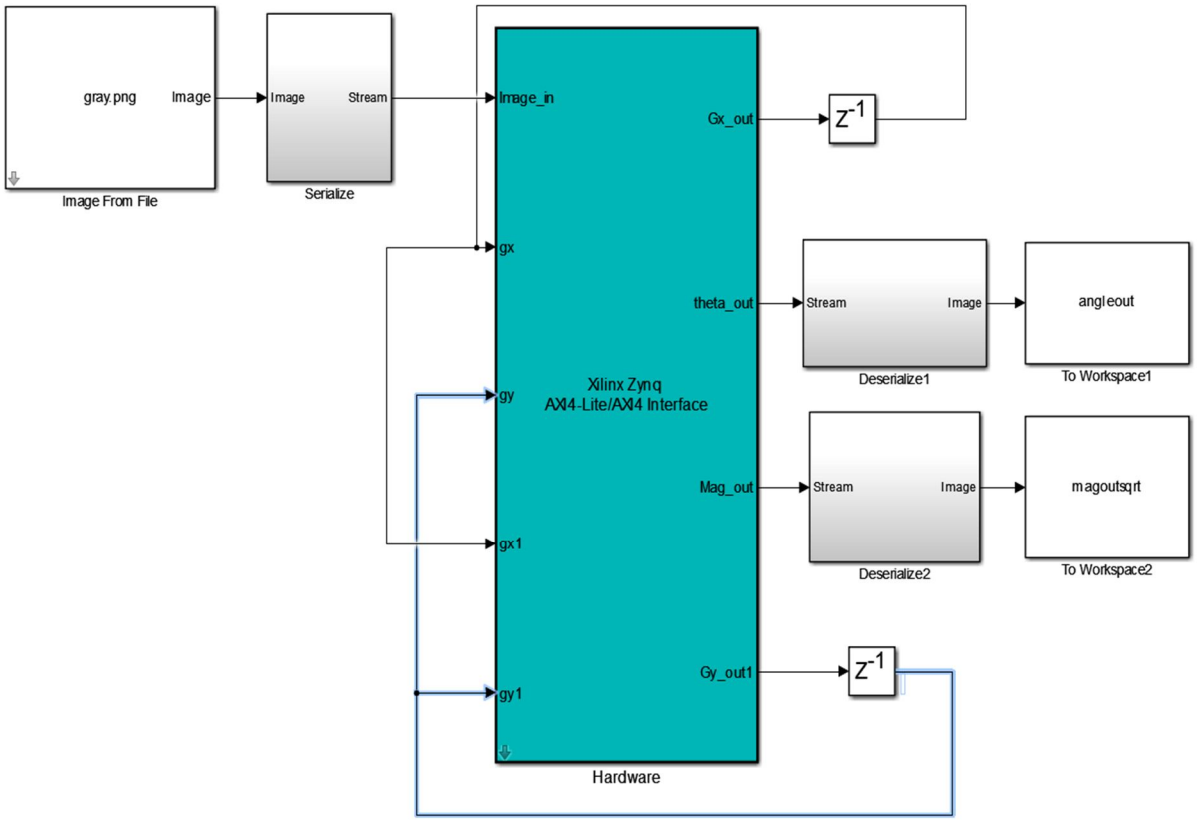


Fig. 12 Simulink model of hardware accelerated pedestrian detection system

Volume 5 Issue IV, April 2017
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

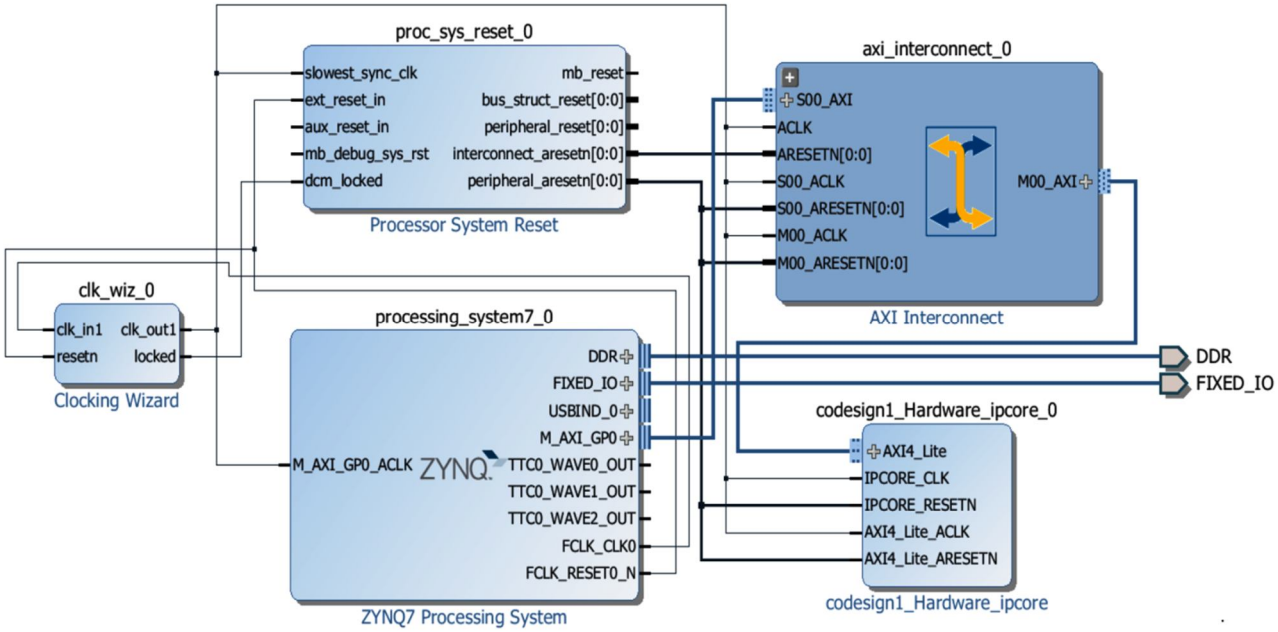


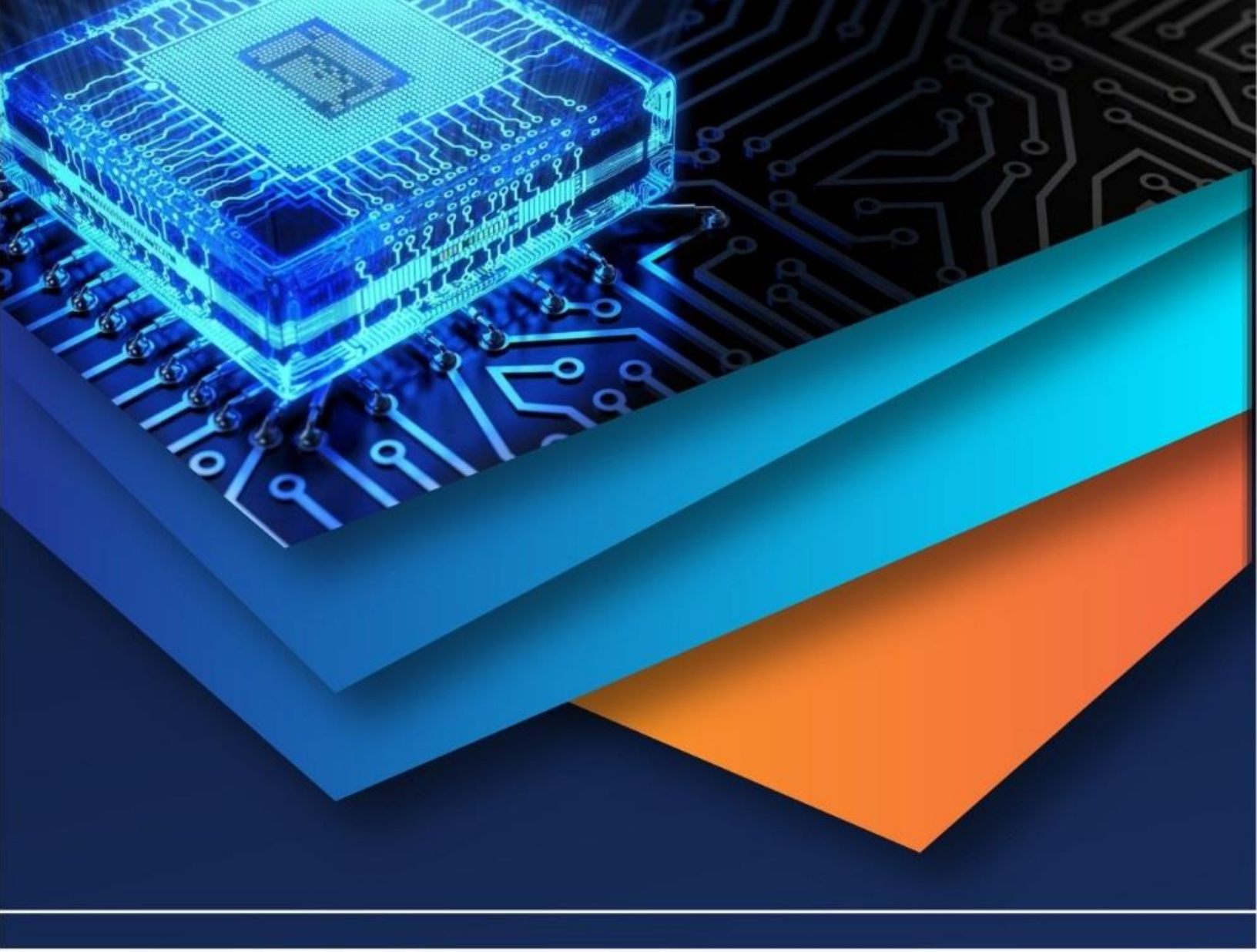
Fig. 13 FPGA design with the IP Core

VI. CONCLUSIONS

The hardware accelerated system was successfully executed using Zed board and Matlab. The accuracy of software simulation as retained while speeding up the system. The system can be further optimised by incorporating the entire system on the FPGA itself. However as the FPGA is fixed point hardware the accuracy will be reduced. Also the same design can be used for high-resolution images with detection windows of 128x64.

REFERENCES

[1] N. Dalal and B. Triggs, “Histograms of Oriented Gradients for Human Detection,” in Proceedings of the 2005 International Conference on Computer Vision and Pattern Recognition, vol. 2, 2005, pp. 886-893.
[2] Jens Rettkowski, Andrew Boutros, Diana Göhringer, “Real-Time Pedestrian Detection on a Xilinx Zynq using the HOG Algorithm," International Conference on Reconfigurable Computing and FPGAs, Dec. 2005.
[3] Luca Maggiani, Cedric Bourrasset, Jean-Charles Quinton, Francois Berry, Jocelyn Serot, “Bio-inspired heterogeneous architecture for real-time pedestrian detection applications”. Journal of Real-Time Image Processing, Springer Verlag, April 2016.
[4] Hong GS, Kim BG, Hwang YS, Kwon KK., “Fast multi-feature pedestrian detection algorithm based on HOG using DWT”. Multimedia Tools and Applications, Springer, Jan. 2015.
[5] Berkant BAŞA., “Implementation of hog edge detection algorithm on fpga’s” In: Elsevier, Procedia - Social and Behavioral Sciences 174, 1567 – 1575, 2015.
[6] V. Campmany, S. Silva, A. Espinosa, J.C.Moure, D. V´azquez, M.L´opez, “GPU-based pedestrian detection for autonomous driving”, In: Elsevier B.V, 2016.
[7] Zhang J, Huang K, Yu Y, Tan T., “Boosted local structured hog-lbp for object localization”. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, IEEE, pp 1393-1400, 2011.
[8] Jun Tanabe, Sano Toru, Yutaka Yamada, Tomoki Watanabe, Mayu Okumura, Manabu Nishiyama, Tadakazu Nomura, Kazushige Oma, Nobuhiro Sato, Moriyasu Banno, Hiroo Hayashi, Takashi Miyamor, “18.2 A 1.9TOPS and 564GOPS/W heterogeneous multicore SoC with color-based object classification accelerator for image-recognition applications”. In: Solid-State Circuits Conference-(ISSCC), 2015 IEEE International, IEEE, pp 1-3, 2015.
[9] Michael Hahnle, Frerk Saxen, Matthias Hisung, Ulrich Brunsmann, Konrad Doll, “FPGA-based Real-Time Pedestrian Detection on High-Resolution Images”. In: IEEE, Conference on Computer Vision and Pattern Recognition Workshops, 2013.
[10] Shuai Xie; Yibin Li; Zhiping Jia; Lei Ju., “Binarization based implementation for real-time human detection”, Field Programmable Logic and Applications (FPL), 2013 23rd International Conference on , vol., no., pp.1,4, 2-4, Sept. 2013.
[11] M. Kachouane, S. Sahki, M. Lakrouf, N. Ouadah, “HOG based fast human detection (Matlab)” In: IEEE 24th International Conference on Microelectronics, 2012.
[12] Hilton Bristow and Simon Lucey, “Why do linear SVMs trained on HOG features perform so well?”. Available at: http://arxiv.org/abs/1406.2419, June 2014.
[13] Support Vector Machines (SVM). MathWorks Documentation Center. http://www.mathworks.com/help/stats/support-vector-machines-svm.html
[14] Support vector machines: The linearly separable case. Stanford University. Cambridge Press. http://nlp.stanford.edu/IR-book/html/htmledition/support-vector-machines-the-linearlyseparable-case-1.html. 2008.
[15] Dan Ventura, SVM Example, April 16, 2009.
[16] INRIA Person Dataset. Available at: http://pascal.inrialpes.fr/data/human



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY