FPGA Based Implementation of Linear SVM for Facial Expression Classification

[1,2] Sumeet Saurav
[1]Academy of Scientific & Innovative Research
[2] CSIR-Central Electronics Engineering Research Institute
Pilani, India
sumeetssaurav@gmail.com

Ravi Saini, Sanjay Singh
IC Design Group
CSIR-Central Electronics Engineering Research Institute
Pilani, India
{ravi, sanjay}.csirceeri@gmail.com

Abstract— **This work presents a Field Programmable Gate Array (FPGA) based hardware efficient implementation of One-Versus-All (OVA) linear Support Vector Machine (SVM) classifier for classifying the facial expressions on an individual. The motivation is to achieve a real-time classification of the facial expressions of an individual into three different states viz., neutral, happy, and pain so that the designed architecture could be used as a part of an embedded platform based FER system for the purpose of monitoring patients in hospitals. Thus, the design challenge is to achieve classification accuracy equivalent to the software-based implementation with a multi-fold improvement in the execution time. The acceleration in the execution time of the designed classifier has been achieved utilizing the parallelism and pipelining concepts of the VLSI architecture design. Moreover, to reduce the computational cost and boost the execution speed of the architecture we have adopted the fixed-point data format (Q24.16) in our design. The classifier has been trained offline and the parameters of the trained classifier have been used to perform testing using the designed architecture on hardware. The designed architecture after synthesis operates at a maximum clock frequency of 241.55 MHz and is resource efficient. Classification accuracy of 98.50% equivalent to its software counterpart has been achieved on simulating the designed architecture with different test images. Thus, the designed classifier architecture shows good performance in terms of speed, area, and accuracy, and is suitable for real-time classification of the facial expressions.**

Keywords— Support Vector Machines, VLSI Architectures, FPGA, AdaBoost, Gabor filter.

I. INTRODUCTION

In today's world of automation, there has been a huge requirement of Facial Expression Recognition (FER) based technology and thus it has become an important area of research among the researchers of the Computer Vision community. The FER technology has a plethora of applications in areas related with human-computer interfaces; patient monitoring, blind person assistance; human emotion analysis [1]; neuroscience, psychology, and cognitive sciences [2]; access control and surveillance [3]; and communication, personality, and child development [4]. Moreover, recent rapid advancement in the Machine Learning and deep learning algorithms has further widened the areas of research in the design of an efficient FER system. However, designing such a system is not a trivial job because of a number of issues and researchers all around the world are trying to mitigate these issues and desired to have human-level performance in the FER based systems.

Over the decades, there have been numerous works reported in the literature dealing with the demonstration of FER system and its practical applications. However, most of these works have targeted towards enhancing the accuracy of these systems without keeping into consideration the real-time constraint which is often the desired requirement.

One of the most important ingredients of any FER system is the classification unit which is used to assign labels to different facial expressions and for this SVM classifier is often considered to the best candidate. Therefore, in this work, we have designed an optimal architecture of an OVA linear SVM classifier to facilitate real-time classification of the facial expressions of an individual.

Most of the available literature on SVM listed in [5]-[12], have discussed their software-based implementation. The first significant work related to the hardware-based implementation of SVM has been reported in [13]. A digital architecture for both the training and testing phase of the SVM using both linear and non-linear (RBF) kernels have been reported. Although, the designed architecture implemented on Xilinx Virtex-II FPGA fulfilled its objectives but suffers from two major drawbacks. Firstly, the FPGA resource utilization is too high and secondly, the processing speed (35.3 MHz) obtained is not acceptable for many applications demanding real-time performance. In order to overcome the issues of high FPGA resource utilization, the authors in [14] proposed a hardware-based SVM classification architecture targeted to FPGA using the Logarithmic Number System (LNS). The authors have claimed to save considerable hardware resources with no significant loss in classification accuracy, but difficulties lie in converting real numbers to their logarithmic equivalent representation. An FPGA friendly implementation of a Gaussian Radial Basis SVM well suited to the classification of grayscale images has been discussed in [15]. The implementation achieved 88.6% detection accuracy (equivalent to software-based implementation) in gender classification. A parallel hardware architecture based FPGA implementation of SVM for the video shot boundary detection application have been discussed in [16]. For brain-computer interface application, an FPGA based implementation of linear kernel support vector machines is reported in [17]. The authors have demonstrated that their hardware model achieved a

classification rate higher than 95% with the fixed-point data format. Incorporating the posterior probability for increasing the performance of the classifier, the authors in [18] have proposed an FPGA implementation of a multi-class support vector machine classifier. For an embedded automotive applications, design of a hardware-friendly support vector machine classifier has been discussed in [19]. In [20], FPGA simulation of linear and non-linear support vector machine has been discussed. Parallelization has been employed in the designed architecture and design has been done using function blocks of the System Generator. According to their simulation results, the maximum frequency of 202.840 MHz in linear classification, and classification accuracy of 98.67% in nonlinear one has been achieved. Support Vector Machine implementation for both classification and regression has been discussed in [21]. In [22], the authors have proposed a hardware architecture of SVM classifier intended for vision applications on FPGA platform. Moreover, with the advent of GPUs (Graphical Processing Units), exploiting its usefulness the authors in [23] have proposed FPGA-GPU based architecture for kernel SVM pedestrian detection. Here, FPGA has been used for feature selection and classification is performed using GPU.

From the algorithmic perspective, SVMs comes under the category of supervised learning algorithms and comprises of a separate training and a test phase. The online training of the SVM classifier is computationally expensive and for applications like FER, it is often not a desirable requirement (as here we are more interested in getting the classification result which comes by performing prediction using the trained SVM model). Therefore, similar to other reported works we also performed SVM classifier training offline in a software environment.

The remaining part of this paper is organized as follows: In section II, we give an introduction and mathematical details of the linear SVM classifier. Section III deals with the software implementation of the algorithm including both training and testing. A detailed description of the proposed hardware architecture is discussed in section IV. In section V simulation and synthesis results have been discussed which is followed by a conclusion in section VI.

II. Overview of Linear SVM Classifier

Support Vector Machines (SVM) is a technique of classification and regression introduced in the 1990s by Vapnik [24]. The naïve implementation of SVM was for binary classification which was later extended for multi-class classification problem. The SVM can be used for classifying linear as well as non-linear data. Since this work is related to linear SVM classification, so all further discussion will be only for linear SVM.

In a linear classification problem it is assumed that the feature set belonging to different classes are linearly separable and the objective is to find an optimal separating hyperplane with maximum margin [25]. The decision function obtained is linear in nature. To understand its concept, let us consider a two class problem consisting of M training samples each having m-dimensional feature set. Let x_i (i=1...M) is m-dimensional training inputs belonging to Class1 or Class2 with the associated labels y_i =1 for Class1 and y_i= -1 for Class2 as shown in Fig. 1.

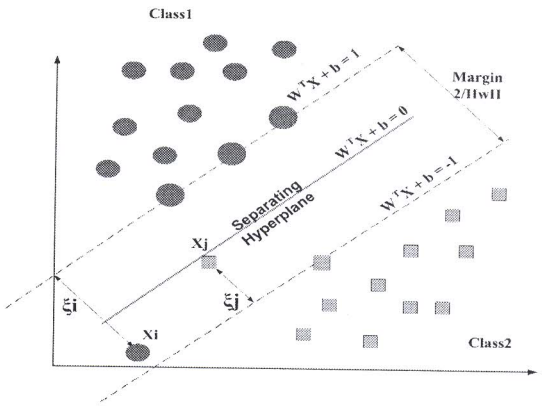

Fig. 1. *Optimal Separating Hyperplane with Maximum Margin and Slack Variables*

The decision function of the linear support vector machine is defined by (1).

$$d(x) = w^T x + b \qquad (1)$$

where, w is an m-dimensional vector, b is a bias term, and for i=1 ...M, the value of the decision function is given by (2).

$$w^T x + b \begin{cases} > 0 \, for \, y_i = 1, \\ < 0 \, for \, y_i = -1 \end{cases} \qquad (2)$$

To control separability at the boundary inequalities of (3) are considered.

$$w^T x + b \begin{cases} \geq 1 \, for \, y_i = 1, \\ \leq -1 \, for \, y_i = -1 \end{cases} \qquad (3)$$

The above equation can be also written as (4).

$$y_i \left(w^T x_i + b \right) \geq 1 \quad \text{For i =1 ...M.} \qquad (4)$$

The hyperplane separating the training data into two classes is given by (5).

$$D(x) = w^T x + b = c \quad for \quad -1 < c < 1 \qquad (5)$$

There are an infinite number of decision functions which satisfy the separating hyperplanes. The optimal separating hyperplane is obtained by solving the dual Lagrange problem of (6).

$$Maximize \; Ld(\alpha) = \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} \alpha_i \alpha_j y_i y_j x_i^T x_j \qquad (6)$$

$$Subject \; to \sum_{i=1}^{M} y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C \quad for \quad i = 1, ...M.$$

Here C, is called the margin parameter that determines the trade-off between the maximization of the margin and the

minimization of the classification error. By solving the dual Lagrange function, α is obtained. Subsequently the value of w and b is also achieved using (7) and (8).

$$w = \sum_{i=1}^{M} \alpha_i y_i x_i \tag{7}$$

$$b = y_i - w^T x_i \quad for \quad i = 1,..S. \tag{8}$$

In (8), S is the set of support vector indices. Now, we can classify an unknown data x, by utilizing (7) and (8) into the decision function of (1), which yield to (9).

$$D(x) = \sum_{i \in S} \alpha_i y_i x_i^T x + b \tag{9}$$

The classification of unknown data x into class 1 or class 2 depending upon the value of decision function D(x) is done as shown in (10). If D(x) = 0, x is on the boundary and thus is unclassifiable

$$x \in \begin{cases} class \;\; 1 \;\; if \;\; D(x) > 0 \\ class \;\; 2 \;\; if \;\; D(x) < 0 \end{cases} \tag{10}$$

As mentioned earlier, the naïve implementation of SVM was for binary classification. Later on it was extended to solve multi-class classification problem using different approaches like One-Versus-All Method, One-Versus-One or Pairwise Classifiers Method, Error-correcting Output Code (ECOC) Method, and All Classes at Once Method [20]. In case of OVA, the number of binary SVM classifier is equal to the number of classes available in the data. However, as shown in the Fig. 2 below, there are some unclassified regions whose class label cannot be determined.

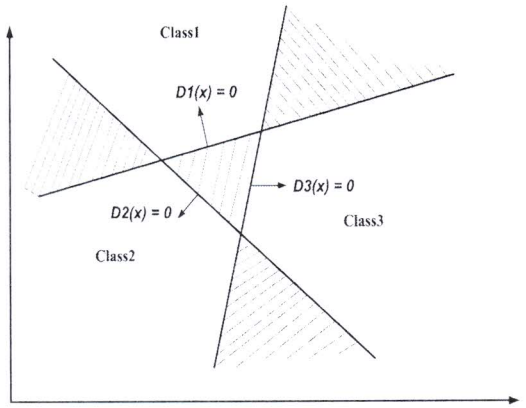

Fig. 2. *Unclassified Regions in OVA Multi-class Classification Approach*

In order to overcome this limitation of the OVA approach, OVO also known as pairwise classifiers approach was proposed by [26]. In this approach to solve n class problem, $\frac{n(n-1)}{2}$ binary classifiers will be formed. This method compared to OVA is better, however there are still some unclassified regions as shown in the Fig. 3. Here, for classifying an input data x, in each binary decision function $d_{ij}(x)$ a class is selected, and finally the one with the most votes is selected as desired class [20].

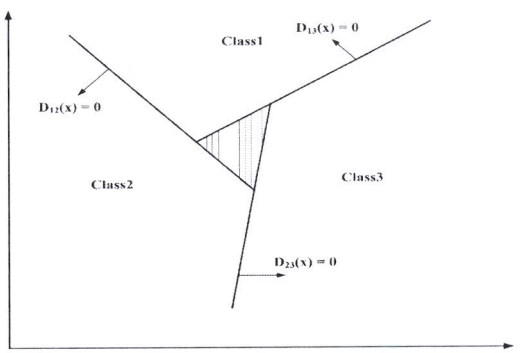

Fig. 3. *Unclassified Regions in OVO Multi-class Classification Approach*

Clearly, there is a trade-off between accuracy and computational resources while making a choice between OVO and OVA multi-class classification strategy. However, depending on the application at hand like FER in our case the classification accuracy does not get that much hindered while opting OVA with an added advantage of reduction in the computational resources. Keeping all these factors into consideration, we opted OVA based multi-class SVM classifier in our work.

III. Software Simulation of OVA Linear SVM

Features required for training and testing the performance of the classifier is obtained from the training and test image samples of three facial expressions by performing a series of steps whose details can be found in our previous work [27]. This basically includes the following sequence of steps: Face detection and registration, feature extraction, and feature selection. For feature extraction we have used Gabor filter. Since the feature obtained from Gabor filters are of high-dimension most of which are redundant, therefore we have used a feature selection scheme using AdaBoost to select optimal features. The extracted features are fed to the SVM classifier which assigns a label to the given facial image. It must be noted here that the performance of classifier depends on the quality of the discriminate features obtained from the feature extraction step.

Distribution of the dataset used in our experiments into training and test samples have been shown in Table I. Here we have used our own database which consists of 56 neutral, 77 happy and 63 pain faces of different individuals whose corresponding cropped sample images is shown in Fig. 4.

TABLE I. Distribution of Training and Test Image Samples

S.No.	Expressions	Training set size	Test set size
1	Neutral	40	16
2	Happy	55	22
3	Pain	40	23

Using the AdaBoost selected features, we performed the classifier training and testing using LibSVM library [28] in the Matlab environment. Here, the objective was to achieve

minimum classification error by varying the value of the margin parameter 'C'. The analysis result of the classifier corresponding to margin parameter C having the maximum classification accuracy is shown in Table II.

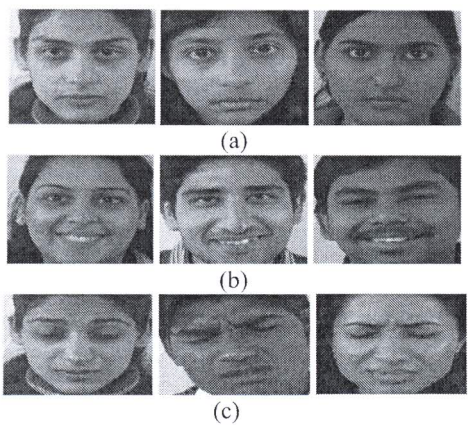

Fig. 4. *Sample Images (a) Neutal (b) Happy (c) Pain*

TABLE II. TRAINED CLASSIFIER PARAMETER FOR C= 0.039

S.No	Class 1	Class 2	Class 3
No. of SVs	29	37	33
Bias	-1.4608	0.6985	-0.4154
Accuracy (%)	98.50		

For our proposed architecture of the testing phase of the OVA linear SVM, we used the parameters obtained from the classifier trained with margin parameter C= 0.039. This is because for this value of the margin parameter the number of SVs is the least and the accuracy is equivalent to the classifier trained with other values of the margin parameter. The confusion matrix corresponding to the selected classifier (with C=0.039) is shown in Table III.

TABLE III. CONFUSION MATRIX OF THE TRAINED CLASSIFIER

	Neutral	Happy	Pain
Neutral	100	0	0
Happy	4.55	95.45	0
Pain	0	0	100
Average	98.50		

IV. PROPOSED ARCHITECTURE OF OVA LINEAR SVM

The block diagram of the proposed architecture of Linear SVM classifier is shown in Fig. 4. Here, we have designed architecture which results in the computation of the decision function given by (9) and the classification function given by (10). The architecture can be divided into three different design units depending on the functionality. The first unit performs inner product operation between the support vectors and test image features. The second part performs dot product operation between the previously computed results the stored αy values. Finally, utilizing the result of the second unit the last unit of the architecture takes a decision regarding the label of the test image. To perform inter and intra coordination of various design units, a controller has been used in the architecture.

A. Inner Product Computation Unit

As shown in Fig. 4, four blocks properly triggered with different control signals are used in the inner product computation unit. These are the support vector storage blocks, input test feature storage block, Kernel Computation block, and Counters.

Three SVs storage blocks have been used in the design for storing SVs of three different classes. These are named as Class1_SVs, Class2_SVs and Class3_SVs corresponding to their respective classes. The blocks have been designed using FPGA's block RAMs. Systematic representation of the Class1_SVs has been shown in Fig. 6. As shown in the figure, we have stored the first 13 columns each of size 51 in SV1_BRAM1, the next 13 columns in SV1_BRAM2 and finally the rest of the 3 columns in SV1_BRAM3. Same techniques have also been used for the storage of SVs for the other two classes. Thus, the depth of each block RAM is 663 (13 × 51) with corresponding width of 24-bits (in Q24.16 format). From the figure, we also find that in the designed architecture the contents of SV1_BRAM1 to SV1_BRAM3 are accessed in parallel with BRAM1_out denoting the output from SV1_BRAM1 and so on. Similar techniques have also been employed for the storage of SVs for the other two classes. Data is read from each block RAM sequentially and the address is given by counter (count663) which is controlled by the controller.

The input test feature storage block represented by Test_Feature in the designed architecture is used for the purpose of storing test features obtained after feature extraction and selection steps. Since the selected feature is of 1×51 dimension, therefore the depth of the block RAM used is 51 with the corresponding width of 24-bits in Q24.16 fixed-point data format. A counter (count51) is used to read the contents of this block.

The kernel computation block as shown in Fig. 7, performs inner product operation between the support vectors and input test features. Three such blocks have been used in the designed architecture (Class1_KC to Class3_KC) corresponding to three different classes. As shown in the figure, the block takes three SVs from the support vector storage block (BRAM1_out1 to BRAM3_out1) and input test feature from the Test_Feature block to perform inner product operation for three different columns of the SVs matrix in parallel. This block consists of three multipliers, three adders, and three accumulators. Each multiplier takes two input, one coming from support vector storage block (BRAM1_out1) and other coming from the Test_Feature block (Input_data). The multiplier output then becomes one input of the adder (ADD1) whose other input comes from the accumulator (ACC1). The final output obtained is of 48-bits denoted by acc1_out1. Similarly, other two outputs from the kernel computation block is acc2_out1 and acc3_out1.

B. Dot Product Computation Unit

As shown in Fig. 4, the dot product computation unit of the designed architecture consists of a temporary storage block, Yalpha value storage block, decision value computation block, some counters. A set of control signals generated by the controller is used to facilitate proper coordination between the blocks.

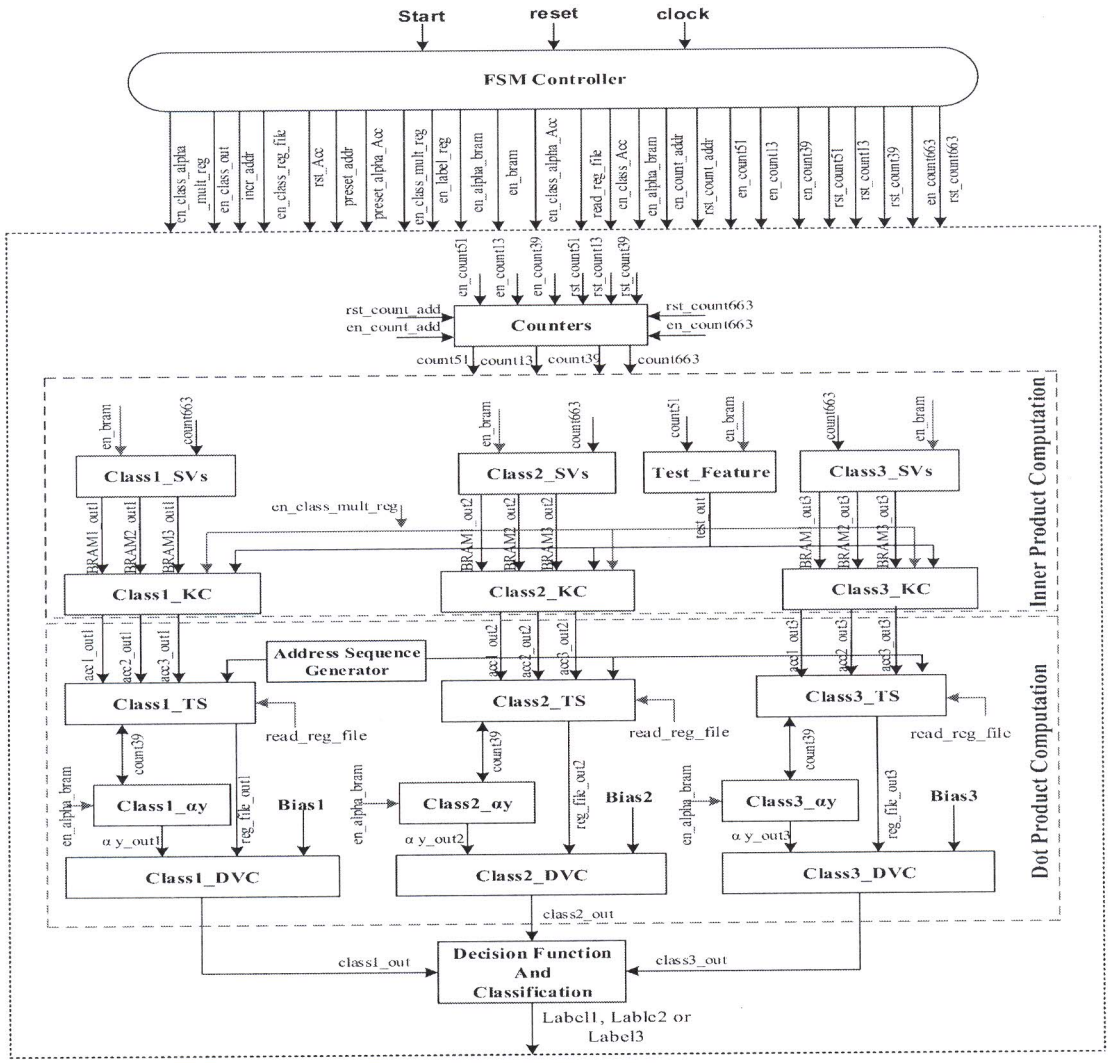

Fig. 5. *Sequential Fixed-Point VLSI Architecture of OVA Linear SVM Classifier*

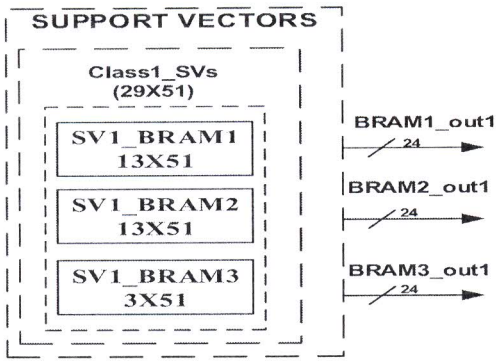

Fig. 6. *Support Vector Storage Block.*

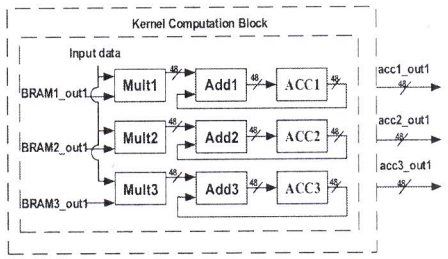

Fig. 7. *Kernel Computation Block*

Temporary storage block is used to store the intermediate results obtained after the kernel computation operation. It

basically consists of a register file with access to read and write data. Since we are performing kernel computation of three classes in parallel, therefore three temporary storage blocks (Class1_TS to Class3_TS) have been used each storing intermediate results from their corresponding kernel computation blocks. Block diagram representation of the temporary storage block has been shown in Fig. 8, where the width of the register file is 39 with a depth of 24-bits. An address sequence generator is used to generate the correct sequence of addresses where the intermediate results need to be stored. The address sequence generator generates three sets of address namely address0, address13, and address26 after every 13 clock cycles which indicates that the inner product of row0, row13 and row26 of SVs matrix with the input test features have been completed and the temporary results need the address for getting stored in the temporary storage block. This operation is repeated with every count of the counter count13. Once the intermediate results have been stores a counter (count39) is used to access the contents of the temporary storage blocks for performing dot product operation with the Yalpha values.

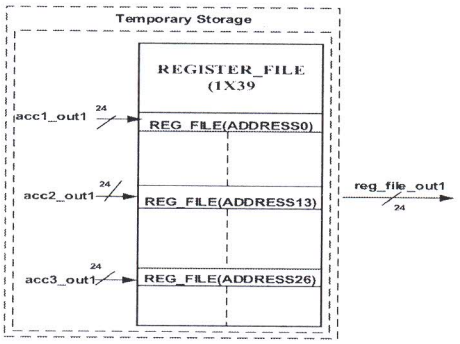

Fig. 8. *Temporary Storage Block.*

Yalpha values corresponding to different classes are also stored in FPGA block RAMs. The width of each block RAM used is 39 with a depth of 24 bits (in the Q24.16 data format). For accessing the value of the block RAMs count39 is used where each count of the counter corresponds to sequential access of one Yalpha value. Three block RAMs have been used in the design corresponding to three different classes.

The final used in the design unit is the Decision Value Computation Block and is shown in Fig. 9. This block performs vector dot product computation between the temporarily stored results of the kernel computation block in the temporary storage block and the Yalpha values stored in the Yalpha storage block. Three such blocks (Class1_DVC to Class3_DVC) have been used in the proposed design corresponding to three different classes. From the figure, it is clear that a 24-bit multiplier has been used to perform multiplication between two inputs (one input coming from the temporary storage block and the other from the Yalpha storage block). The output of the multiplier is given as input to the 48-bit adder whose another input comes from the accumulator ALPHA_ACC1 which initially contains bias value Bias1 which is numerically equivalent to the addition of 'b' as shown in (9). The outputs obtained from each of the three decision value computation block are of 24-bits denoted by class1_out, class2_out, and class3_out respectively.

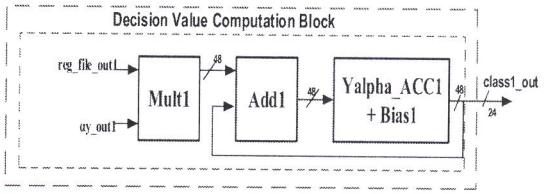

Fig. 9. *Decision Value Computation Block.*

C. Classification Unit

The final unit called classification unit consists of Decision function and Classification block. It is used in the designed architecture to decide the label of the input test feature and provide the corresponding label (either label1, label2, or label3) as output. This block takes three inputs coming from the three decision value computation blocks and gives output as label1, label2, and label3. Decision using this block is made as follows: label1 is selected when the value of class1_out is positive and other two (class2_out and class3_out) is negative, label2 is selected when class2_out is positive and other two (class1_out and class3_out) is negative, label3 is selected when class3_out is positive and other two (class1_out and class2_out) is negative. Finally, one of the labels among label1, label2, and label3 will be the output of our proposed linear SVM classifier.

D. Controller Block

For executing a proper sequence of operations and enabling coordination between different blocks, a controller (Finite-State Machine) has been used in the designed architecture. The controller generates control signals for enabling different counters, accumulators and other storage elements used in the design at a proper instant of time. The state diagram representation of the controller is shown in Fig. 10.

V. Simulation And Synthesis Results

All the modules of the proposed architecture are coded in VHDL and simulated using ModelSim 10.1C. The simulation result has been shown in Fig. 11. From the figure, we find that the given input test image belongs to class 3 (since the value of label3=1). Synthesis is carried out using Xilinx ISE tool chain (version 14.2). We have used Xilinx ML510 (Virtex-5 FXT) FPGA platform for synthesizing the design. Fixed point numbers is used for quantization and Q24.16 quantization format is used. Table IV shows a comparison of FPGA simulation results of the proposed linear SVM classifier with its software implementation in MATLAB. Rows of 1 through 3 shows the confusion matrix of the hardware and software classification results. From Table IV, we find that the designed architecture achieves the classification accuracy similar to that of its software version, but its execution time is much less, which facilitates real-time classification of the facial expressions. The hardware resources utilized by the design have been listed in Table V.

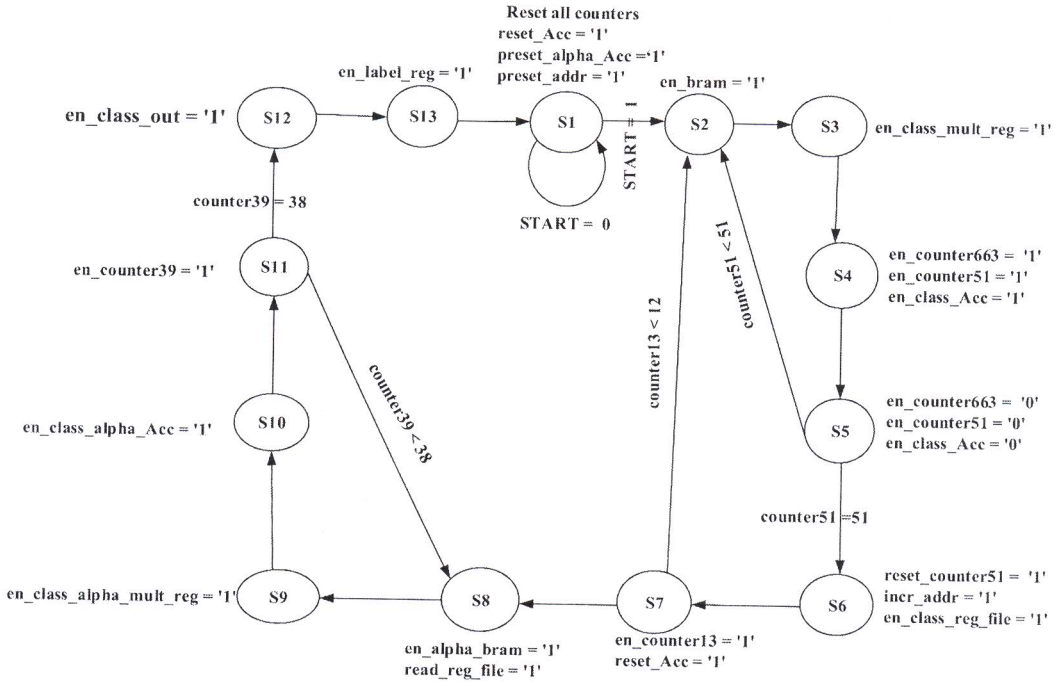

Fig. 10. *Controller State Diagram.*

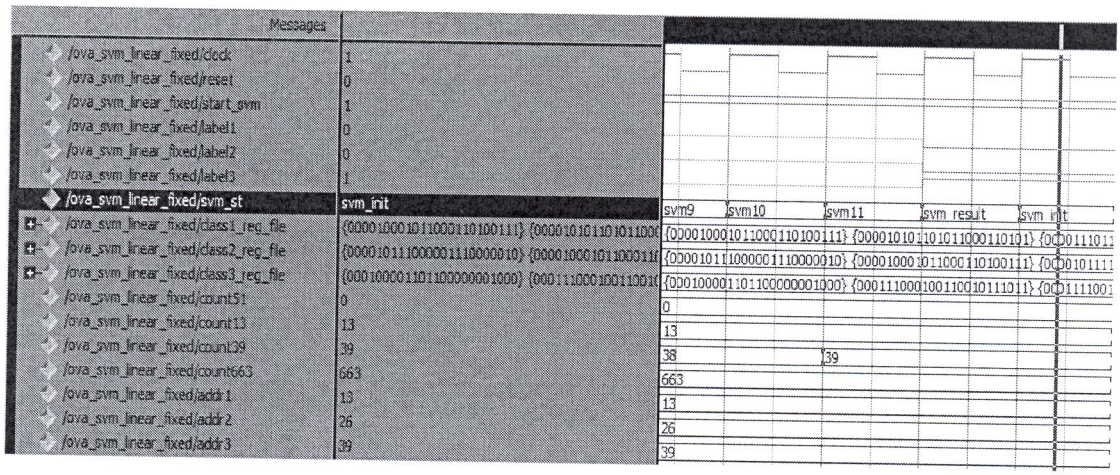

Fig. 11. *Simulation Result of the Designed Archite*

VI. Conclusion

In this paper, hardware efficient FPGA based implementation of OVA linear SVM classifier for facial expression classification has been presented. The designed architecture has been coded in VHDL and implemented on Xilinx ML510 FPGA platform. The implemented architecture can perform real-time facial expression classification operating at a clock frequency of 241.55 MHz with a recognition accuracy of 98.50%. The proposed architecture outperforms the reference architecture [26] and [20], in terms of speed, and area without compromising the accuracy. Extending the architecture for more facial expression classes and integrating it with other blocks used in FER system is the purpose of our future work. Moreover, we want to point out that the designed architecture is not generic, but one can design the similar architecture for their own application.

TABLE IV. COMPARISION OF HARDWARE AND SOFTWARE SIMULATION RESULTS

	VHDL			MATLAB		
	Class1	Class2	Class3	Class1	Class2	Class3
Class1	16	0	0	16	0	0
Class2	1	21	0	1	21	0
Class3	0	0	23	0	0	23
Error Rate	1.64 %			1.64 %		
Number of misclassified compared with MATLAB	0			-		
Maximum Frequency	241.546 MHz, Virtex-V			2.17 GHz, Intel Cor2Duo		
Time of Computation	4.140 ns			65 ms		

TABLE V. HARDWARE UTILIZATION RESULT OF THE PROPOSED ARCHITECTURE

Logic Utilization	Available	Used	Utilization (%)
Slice FFs	81920	6734	8
Slice LUTS	81920	6511	7
Occupied Slice	20480	2026	9
Bonded IOB	840	75	7
BUFG	32	2	6
RAMB36	298	4	1
DSP48	320	24	7

Acknowledgment

The authors express their deep sense of gratitude to Prof. Santanu Chaudhary, Director CSIR-CEERI for encouraging research and development activities. Authors would also like to thanks Dr. A.S Mandal, Group Leader, Cognitive Computing Group, CSIR-CEERI, for his constructive suggestions. The financial support of DeitY/MCIT is also gratefully acknowledged.

REFERENCES

[1] Fasel, B.; Luettin, J. Automatic facial expression analysis: A survey. Pattern Recognition. 2003, 36, 259-275.

[2] Bettadapura, V. Face Expression Recognition and Analysis: The State of the Art, Tech Report arXiv: 1203.6722; April 2012; pp. 1-27.

[3] Wu, X; Zhao, J. Curvelet Feature Extraction for Face Recognition and Facial Expression Recognition. In Proceedings of IEEE 6th International Conference on Natural Computation, Yantai, China, 10-12 August 2010; Volume 3, pp. 1212-1216.

[4] Bartlett, M; Hager, J; Ekman, P; Sejnowski, T. Measuring facial expressions by computer image analysis. Psychophysiology 1999, 36, 253-263.

[5] Hirata, W; Tan, J.K; Kim, H; Ishikawa, S. Recognizing facial expression for man-machine interaction. In Proceedings of IEEE ICCAS-SICE, Fukuoka, 18-21 August 2009; pp. 1621-1624.

[6] Beszedes, M; Culverhouse, P; Oravec, M. Facial emotion classification using active appearance model and support vector machine classifier. Machine Graphics and Vision International Journal, 2009, 18, 21-46.

[7] Kotsia, I; Pitas, I. Facial expression recognition in image sequences using geometric deformation features and support vector machines. IEEE Trans. Image Process. 2007, 16, 172-187.

[8] Kotsia, I; Pitas, I. Real time facial expression recognition from image sequences using support vector machines. In Proceedings of IEEE International Conference on Image Processing, 11-14 September 2005; Volume 2, pp. II-966-9.

[9] Dumas, M. Emotional expression recognition using support vector machines. In Proceedings of International Conference on Multimodal Interfaces, 2001.

[10] Tsai, H.H; Lai, Y.S; Zhang, Y.C. Using SVM to design facial expression recognition for shape and texture features. In Proceedings of IEEE International Conference on Machine Learning and Cybernetics, Qingdao, 11-14 July 2010; Volume 5, pp. 2697-2704.

[11] Patil, R.A; Sahula, V; Mandal, A.S. Feature classification using support vector machine for a facial expression recognition system. *Journal of Electronic Imaging*, 2012, 21, 043003-1.

[12] Visutsak, P. Emotion Classification through Lower Facial Expressions using Adaptive Support Vector Machines. *Journal of Man, Machine and Technology*, 2013, 2, 12-20.

[13] Anguita, D; Boni, A; Ridella, S. A digital architecture for support vector machines: theory, algorithm, and FPGA implementation. *IEEE Trans. Neural Networks*. 2003, 14, 993-1009.

[14] Khan, F.M; Arnold, M.G; Pottenger, W.M. Hardware-based support vector machine classification in logarithmic number systems. In Proceedings of IEEE International Symposium on circuits and systems, 23-26 May 2005; Volume 5, pp. 5154-5157.

[15] Irick, K.M; DeBole, M; Narayanan, V; Gayasen, A. A hardware efficient support vector machine architecture for FPGA. In Proceedings of IEEE 16th International Symposium on Field-Programmable Custom Computing Machines, Palo Alto, CA, 14-15 April 2008; pp. 304-305.

[16] Hsu, C.F; Ku, M.K; Liu, L.Y. Support vector machine FPGA implementation for video shot boundary detection application. In Proceedings of IEEE International SOC Conference, Belfast, 9-11 September 2009; pp. 239-242.

[17] Pina-Ramirez, O; Valdes-Cristerna, R; Yanez-Suarez, O. An FPGA implementation of linear kernel support vector machines. In Proceedings of IEEE International Conference on Reconfigurable Computing and FPGA's, San Luis Potosi, 20-22 September 2006; pp. 1-6.

[18] Nie, Z; Zhang, X; Yang, Z. An FPGA Implementation of Multi-Class Support Vector Machine Classifier Based on Posterior Probability. In Proceedings of 2010 3rd International Conference on Computer and Electrical Engineering, 2012.

[19] Anguita, D; Ghio, A; Pischiutta, S; Ridella, S. A Hardware-friendly Support Vector Machine for Embedded Automotive Applications. In Proceedings of IEEE International Joint Conference on Neural Networks, Orlando, FL, 12-17 August 2007; pp. 1360-1364.

[20] Mahmoodi, D; Soleimani, A; Khosravi, H; Taghizadeh, M. FPGA simulation of linear and nonlinear support vector machine. *Journal of Software Engineering and Applications*, 2011, 4, 320.

[21] Ruiz-Lita, M; Yebenes-Calvino, M. FPGA implementation of a support vector machine for classification and regression. In Proceedings of the IEEE International Joint Conference on Neural Networks, Barcelona, 18-23 July 2010; pp. 1-5.

[22] Ruiz-Liata, M; Yebenes-Calvino, M. FPGA implementation of support vector machines for 3D object identification. In *Artificial Neural Networks-ICANN 2009*; Springer: Limassol, Cyprus, 14-17 September 2009; pp. 467-474.

[23] Bauer, S; Kohler, S; Doll, K; Brunsmann, U. FPGA-GPU architecture for kernel SVM pedestrian detection. In Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, San Francisco, CA, 13-18 June 2010; pp. 61-68.

[24] Vapnik, V. Statistical Learning Theory. New York: John Wiley & Sons Inc. 1998.

[25] Abe, S. Support vector machines for pattern classification. London: Springer. 2005, 2.

[26] Kreßel, U.H.G., 1999, February. Pairwise classification and support vector machines. In Advances in kernel methods (pp. 255-268). MIT press.

[27] Saurav, S., Singh, S., Saini, R., & Saini, A. K. (2016). Hardware Accelerator for Facial Expression Classification Using Linear SVM. In Advances in Signal Processing and Intelligent Recognition Systems (pp. 39-50). Springer International Publishing.

[28] Chang, C.C; Lin, C.J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*. **2011**, 2, 27.