VLSI Architecture of Pairwise Linear SVM for Facial Expression Recognition

Sumeet Saurav, Anil K Saini, Sanjay Singh, Ravi Saini
IC Design Group
CSIR-Central Electronics Engineering Research Institute
Pilani, Rajasthan, India
sumeetssaurav@gmail.com

Shradha Gupta
AIM & ACT
Banasthali Vidhyapeeth
Tonk, Rajasthan, India

Abstract— **In this paper, we present VLSI architecture of Pairwise Linear Support Vector Machine (SVM) classifier for multi-classification on FPGA. The objective of this work is to facilitate real time classification of the facial expressions into three categories: neutral, happy and pain, which could be used in a typical patient monitoring system. Thus, the challenge here is to achieve good performance without compromising the accuracy of the classifier. In order to achieve good performance pipelining and parallelism (key methodologies for improving the performance/frame rates) have been utilized in our architectures. We have used pairwise SVM classifier because of its greater accuracy and architectural simplicity. The architectures has been designed using fixed-point data format. Training phase of the SVM is performed offline, and the extracted parameters have been used to implement testing phase of the SVM on the hardware. According to simulation results, maximum frequency of 241.55 MHz, and classification accuracy of 97.87% has been achieved, which shows a good performance of our proposed architecture.**

Keywords—Support Vector Machines; Pairwise SVM; VLSI Architectures; Classification; LibSVM.

I. INTRODUCTION

Facial expression is one of the most powerful and immediate means for humans to communicate their emotions, cognitive states, intensions, and opinions to each other [1]. Recent advances in the Computer Vision algorithms equipped with Machine Learning(Pattern Classification) algorithms has open up wide areas of research in the design of automatic Vision systems deploying a blend of these algorithms like Facial Expression recognition system, Object detection system etc. One of the most important block which is used in almost all facial expression recognition system is classification using support vector machines (SVM).Thus, the objective of this work is to facilitate classification using dedicated hardware architectures.

Most of the available literature on SVM is based on software i.e. both the training and testing phase of the algorithm are done on software [2]-[9]. However, in terms of hardware implementation, the first significant work on SVM has been reported in [10]. In this work, the authors have proposed a digital architecture for SVM training and classification using linear and RBF kernels on Xilinx Virtex-II FPGA. The major drawback of this implementation is that it utilized a lot of resources. Followed by this, over the time a number of works has also been reported in literature dealing with efficient hardware implementation of support vector machine classifiers. These works can be found in [11]-[20].

Keeping in view the hardware constraints involved in the training phase of the SVM, like earlier reported works we have also performed this phase on software. This is a reasonable choice as far as application is concerned, since we want the classification result which comes from the testing phase of the SVM, utilizing the parameters obtained after the training phase. Thus, in order to attain real-time performance of our application we have done only the testing phase of the SVM algorithm on hardware and have utilized various capabilities of the FPGA (parallelism and pipelining) to achieve better performance. To optimize the resources we have designed the architecture using fixed-point data format (Q24.16).

The remainder of the paper is organized as follows: In section II, software implementation of the algorithm is described. Detailed description of the proposed hardware architecture is discussed in section III. In section IV, simulation results has been discussed which is followed by conclusion in the section V.

II. SOFTWARE IMPLEMENTATION OF PAIRWISE LINEAR SVM

Before hardware implementation of the SVM architecture, we performed both training and testing of the algorithm in software environment using MATLAB. The target is to get the lowest classification error rate by changing various classifier parameters. The parameters necessary for the classification using our proposed VLSI architecture are extracted from the classifier trained with the lowest classification error.

In order to generate the facial features we performed face detection using Viola and Jones face detection algorithm for both training and testing image samples. The detected faces are cropped and resized to 128 x128 size. On the resized faces Gabor filtering is applied [21] whose spatial domain representation is shown in (1). In the spatial domain, a two-dimensional Gabor filter is a Gaussian kernel function modulated by a complex sinusoidal plane wave.

$$G(x,y) = \frac{f^2}{\pi\gamma\eta}\exp\left(-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}\right)\exp\left(j2\pi f x' + \phi\right)$$

where

$$x' = x\cos(\theta) + y\sin(\theta) \qquad (1)$$
$$y' = -x\sin(\theta) + y\cos(\theta)$$

In (1), f is the frequency of the sinusoidal factor, θ represents the orientation of the normal to the parallel strips of a Gabor function, ϕ is the phase effect, σ is the standard deviation of the Gaussian envelope and γ is the spatial aspect ratio which specifies the ellipticity of the support of the Gabor function. We have employed forty Gabor filters in five scales and eight orientations. Here the size of the output feature vector is the size of the image (128x128) multiplied by the number of scales and orientations (5x8) divided by the row and column down-sampling factor (4x4) which is 128x128x5x8/(4x4) =40960 in total. The feature vector is still large even after down-sampling. Therefore, we need to use dimensionality reduction or feature selection methods to reduce the feature size [22]. Feature selection using AdaBoost [23] has been performed on the Gabor filter generated features. This significantly reduced the feature size from 40960 to 51 without significantly affecting the performance of the classifier.

We have used our own database which consists of 56 neutral, 77 happy and 49 pain faces of different individual for the purpose of training and testing. The distribution of the database into train and test sets has been shown in table I and sample images are shown in the fig 1.

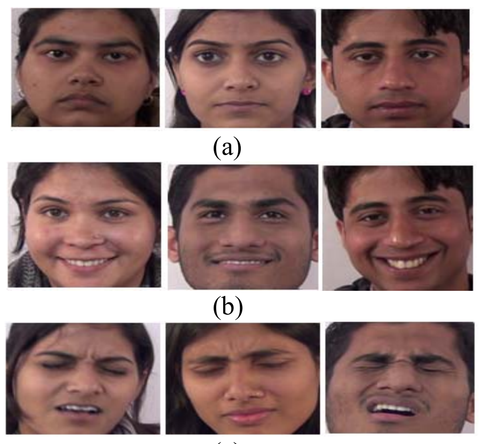


Fig. 1. Training samples of the three classes used in our system: (a) Neutral (b) Happy (c) Pain.

TABLE I. DISTRIBUTION OF TRAINING AND TEST IMAGES

S.No.	*Expressions*	*Training set size*	*Test set size*
1	Neutral	40	16
2	Happy	55	22
3	Pain	40	09

A. *Pairwise Classification using LibSVM with AdaBoost Reduced Feature Set*

Pairwise classification on AdaBoost reduced dataset has been done with LibSVM. The variation of classifier accuracy with the margin parameter 'C' is shown in table II.

TABLE II. VARIATION OF CLASSIFIER ACCURACY WITH MARGIN PARAMETER C

S.No.	**C**	**Accuracy (%)**
1	0.01	97.8723
2	0.02	97.8723
3	0.03	97.8723
4	0.04	97.8723
4	0.06	93.6170
5	1	91.4894

The classifier parameters corresponding to different values of the margin parameter C has been shown in table III-table VI.

TABLE III. CLASSIFIER PARAMETER CORRESPONDING TO THE VALUE OF C=0.01

S.No	**Class1 & 2**	**Class1 & 3**	**Class2 & 3**
	Class1 Class2	***Class1 Class3***	***Class2 Class3***
No. of SVs	24 27	24 28	27 28
Bias	0.3664	0.7152	0.3249

TABLE IV. CLASSIFIER PARAMETER CORRESPONDING TO THE VALUE OF C=0.02

S.No	**Class1 & 2**	**Class1 & 3**	**Class2 & 3**
	Class1 Class2	***Class1 Class3***	***Class2 Class3***
No. of SVs	22 23	22 21	23 21
Bias	0.6373	0.8112	0.4268

TABLE V. CLASSIFIER PARAMETER CORRESPONDING TO THE VALUE OF C=0.03

S.No	**Class1 & 2**	**Class1 & 3**	**Class2 & 3**
	Class1 Class2	***Class1 Class3***	***Class2 Class3***
No. of SVs	18 19	18 20	19 20
Bias	0.7077	0.7948	0.4500

TABLE VI. CLASSIFIER PARAMETER CORRESPONDING TO THE VALUE OF C=0.04

S.No	Class1 & 2	Class1 & 3	Class2 & 3
	Class1 Class2	*Class1 Class3*	*Class2 Class3*
No. of SVs	17 19	17 18	19 18
Bias	0.7022	0.8337	0.4124

From, the hardware point of view we need classifier with similar performance but with less support vectors, as less support vectors requires less storage and there will less computation. Therefore, we selected the parameters corresponding to the classifier trained with the margin parameter C=0.04, since it gives lesser number of support vectors as compared to the classifiers trained with other values of the margin parameter C. The confusion matrix corresponding to the selected classifier (with C=0.04) is shown in table VII. We have used AdaBoost reduced feature set for our hardware implementation, since in this case we only need to store the indices corresponding to the best features to be used for selecting features in a new test image.

TABLE VII. CONFUSION MATRIX CORRESPONDING TO CLASSIFIER TRAINED WITH C=0.04

	Neutral	Happy	Pain
Neutral	16	0	0
Happy	1	21	0
Pain	0	0	9

III. PROPOSED VLSI ARCHITECTURE OF PAIRWISE LINEAR SVM

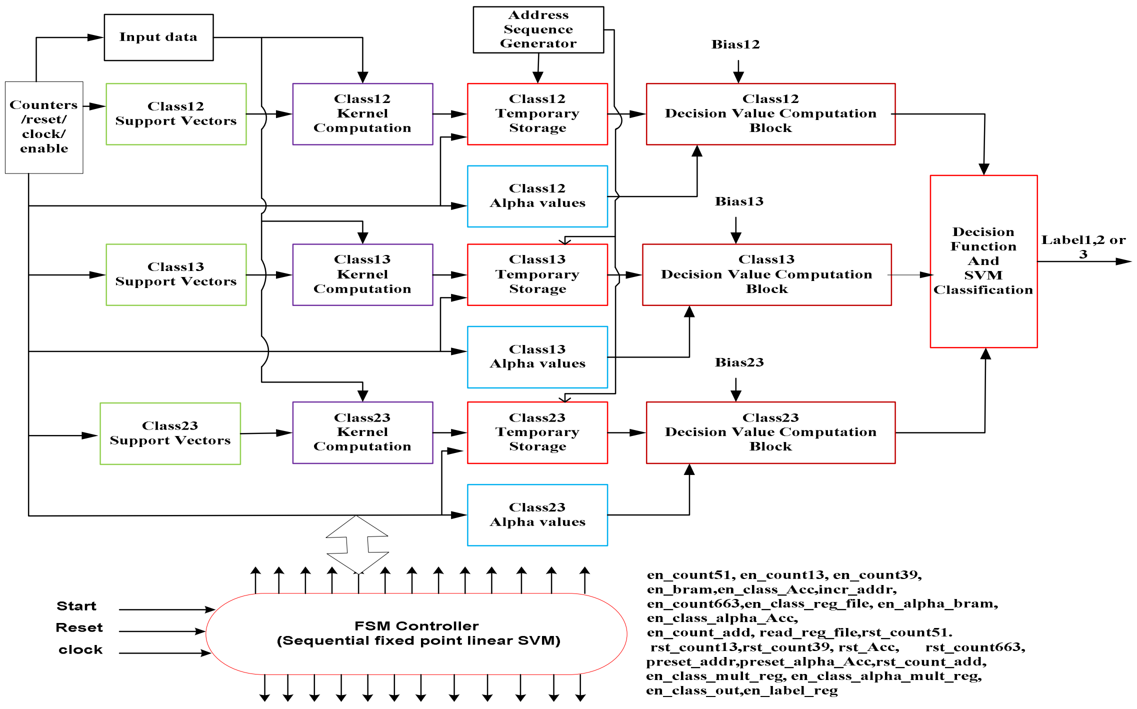
The block diagram of the proposed VLSI architecture of pairwise linear Support Vector Machine is shown in fig. 2. Here, we have developed a proper set of blocks which results in the computation of the decision function (2). The kernel computation between different classes is performed using their respective support vectors and the input data.

$$D(x) = \sum_{i \in S} \alpha_i y_i x_i^T x + b \qquad (2)$$

Temporary storage is required to store intermediate results which are then used for further computation. All alpha values operations are performed in Decision Value computation block. After performing all these operations for class12, class13 and class23, prediction of labels (label1, label2, label3) is done in Decision function and SVM classification block. The execution of all these blocks are controlled by a FSM controller which provides proper control signals at proper interval of time. Detailed description of all of these blocks is described below.

A. Input Data Storage Block

The size of input data (feature vector obtained after feature selection) is 1x51 and these are stored in the FPGA block memory. One block RAM is used to store the complete input data. Each feature vector of the input data is of 24 bits in (Q24.16) format**,** where 8 bits are used for integral part and 16 bits for fractional part.



Input data
Address Sequence Generator
Bias12
Counters /reset/ clock/ enable
Class12 Support Vectors
Class12 Kernel Computation
Class12 Temporary Storage
Class12 Decision Value Computation Block
Class12 Alpha values
Bias13
Decision Function And SVM Classification
Label1,2 or 3
Class13 Support Vectors
Class13 Kernel Computation
Class13 Temporary Storage
Class13 Decision Value Computation Block
Class13 Alpha values
Bias23
Class23 Support Vectors
Class23 Kernel Computation
Class23 Temporary Storage
Class23 Decision Value Computation Block
Class23 Alpha values
Start
Reset
clock
FSM Controller (Sequential fixed point linear SVM)
en_count51, en_count13, en_count39, en_bram,en_class_Acc,incr_addr, en_count663,en_class_reg_file, en_alpha_bram, en_class_alpha_Acc, en_count_add, read_reg_file,rst_count51. rst_count13,rst_count39, rst_Acc, rst_count663, preset_addr,preset_alpha_Acc,rst_count_add, en_class_mult_reg, en_class_alpha_mult_reg, en_class_out,en_label_reg

Fig. 2. Sequential fixed point VLSI architecture for linear Pairwise SVM

B. Support Vector Storage Block

This block shown in Fig. 6, is used to store support vectors obtained after SVM training, using block RAMs available in FPGA. The width of each block RAM used is 663 (storing 13 rows and 51 columns of support vector matrix) and the corresponding depth is 24 bits (in Q24.16 format). Here we have used 3 block RAMs (SV12_BRAM1 to SV12_BRAM3) for class12, another 3 block RAMs (SV13_BRAM1 to SV13_BRAM3) for class13 and yet another 3 block RAMs (SV23_BRAM1 to SV23_BRAM3) for class23 to store all values of support vector matrix. Data is read from each block RAM sequentially and the address is given by counter (counter663) which is controlled by the controller. A total 9 block RAMs are needed to store support vectors of all the three classes as shown in fig. 3.

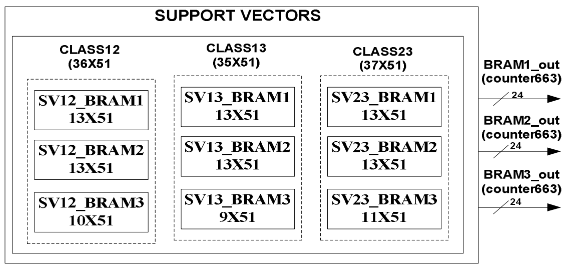


Fig. 3. Support Vectors Storage Block

C. YAlpha Values Storage Block

Similar to support vector 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 8.16 data format). Data is read from each block RAM sequentially and the address is given by counter (counter39). Three block RAMs (YALPHA12_BRAM to YALPHA23_BRAM) are used for each class to store the Yalpha (αy) values as shown in fig 4. Size of Yalpha values for class12, class13 and class23 are 1x36, 1x35 and 1x37 respectively.

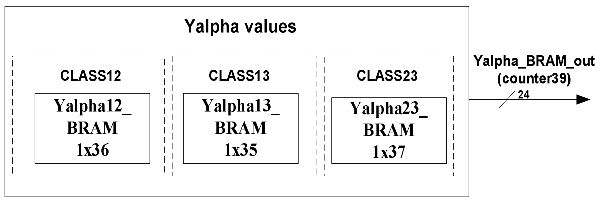


Fig. 4. Yalpha Value Storage Block

D. Temporary Storage Block

Temporary storage block consists of three register files to store the accumulator value after kernel computation for each classes. The size of each register file is 39 with a depth of 24 bits as shown in fig. 5. Addresses for the register file is generated by the address sequence generator block. Since three kernel computation units are working in parallel for each classes, therefore the address generator block generates three addresses in parallel to access three registers from each register files corresponding to all three classes at a time. It generates 3 addresses namely address0, address13 and address26 for a register file at a fixed interval of 13 clock cycles. Initially address0 points to the 0th location of register file, address13 points to the 13th location of register file and address 26 points to the 26th location. When counter13 is incremented, all the addresses are also incremented after storing the intermediate results of first row.

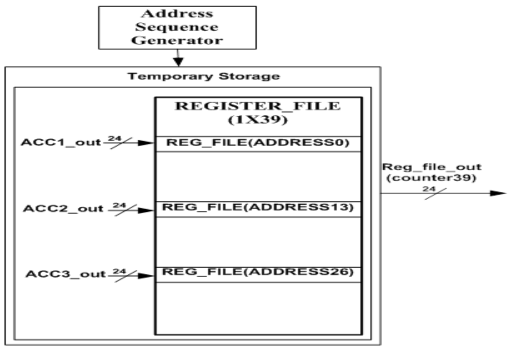


Fig. 5. Temporary Storage Block.

E. Counters Block

Counters are used to give address to the different storage blocks used in the design discussed above. Four different counters have been used in the design each used to access data from different data storage blocks. The Counter51 is used to count 51 support vectors stored in the SV_BRAMs (1 row of support vector matrix). This counter is also used to give address to the input data storage block. When this counter count becomes 51, it indicates that all 51 support vectors corresponding to one row of the support vector matrix and also one row of the input data matrix has been processed. After complete processing of the first row of the support vectors and input data, this counter is reset to process the next row of support vectors and this process is repeated till all the support vectors and input data has been processed. Since 13 rows of the support vectors are stored in one block RAM so a counter called Counter13 is used to provide control signals to controller which in turn causes access of support vectors from the other support vector block RAMs. When counter51 count becomes 51, this counter is incremented by one indicating that 1 row is completed. When this counter count becomes 13, it indicates that processing of all 13 rows is completed and processing starts with another set of SV_BRAMs. A global counter, denoted by Counter663 to give address to all the block RAMs. Since the size of each block RAM is 663, therefore each count provides access to one support vector each clock cycle. To access the contents of the register files and Yalpha block RAMs, another counter (Counter39) has been used. To give address to all values stored in the register files and alpha block RAMs. When the count of this counter becomes 39, it indicates that the decision value of each class has been calculated which are then fed to the Decision Function and SVM classification block.

F. Kernel Computation Block

Once the support vector and input data is read from their respective storage block, kernel computation takes place in the Kernel Computation block. This block is used to perform vector dot product between input data and transpose of the support vectors. We have used three such blocks corresponding to each classes in our design. Systematic representation of the operation performed by this block has been shown in fig. 6 and a brief description of the functionality performed by this block is described below.

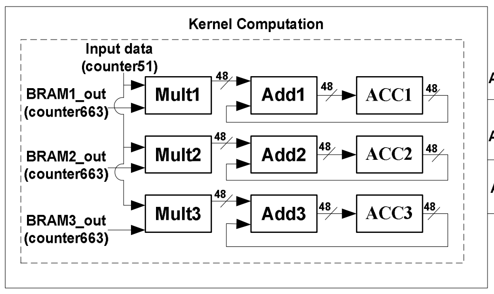


Fig. 6. Kernel Computation Block.

As shown in the figure , a set of three 24-bit multipliers are used to perform multiplication operation of two 24-bit wide fixed point numbers (one being the input data addressed by counter51 and other is support vector addressed by counter663). For the kernel computation of all classes (3 in our case) a total of 9 multipliers (3 for class12, 3 for class13 and 3 for class23) are used and multiplication is performed in one clock cycle. Output from the multiplier is given to their respective adders which are also of 24-bit. One input of the adder comes from the multiplier unit and the other form the accumulator. Similar to the multipliers a total of 9 adders (3 for class12, 3 for class13 and 3 for class23) has been used in the design which also perform addition in one clock cycle. The third block used in the kernel computation unit is the accumulator block which is basically a register of 48-bit. This block is used to accumulate all the values which are resulted after addition operation. Here we are using 3 accumulators (ACC1 to ACC3) for class12, another 3 accumulators for class13 and yet another 3 for class23. So, a total of 9 accumulators have been used for all the three classes. The value of these accumulators are reset to zero when the count of the counter51 becomes equal to 51 which indicates completion of dot product operation between first rows of the input data with that of the support vectors.

G. Decision Value Computation Block

Once dot product computation between the support vectors and input data vectors gets completed and the computed values gets stored in the register file, decision value computation between the Yalpha values and contents of the register files is started. Block diagram description of the decision value computation block is shown in fig. 7. As shown in the figure, a 24-bit multiplier accepts two inputs one from the register file and other from the Yaplha_BRAM. Address to both of these storage elements is provided by the counter39. The output of the multiplier is given as input to the 48-bit adder who’s another input comes from the accumulator ALPHA_ACC12. Initially, the accumulator is preset with bias value of Bias12 which is equivalent to addition of 'b' as shown in the equation (11). Three such Yalpha computation blocks has been used in our design corresponding to three all classes.

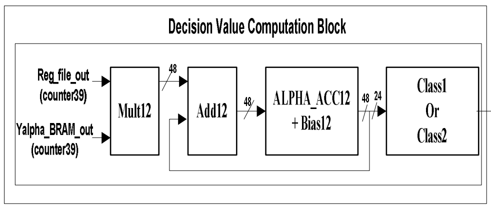


Fig. 7. Decision Value Computation Block.

H. Decision Function and SVM Classification Block

This is the final block in our design which takes output from three Yalpha computation block and decides the label of the input test data and provides it the corresponding label (either label1, label2, or label3). Decision is made as follows: When the value of class12_out and class13_out both is positive and that of class23_out is negative, then label1 is assigned to the input test data, label2 is selected when class12_out and class23_out is positive and class13_out is negative, and similarly label3 is selected when class13_out and class23_out are positive and class12_out is negative. Finally, one of the labels among label1, label2 and label3 will be the output of our proposed Pairwise Linear Support Vector machine classifier.

I. Controller Block

Coordination between different blocks discussed above is done with the help of a controller. The top level description of the controller used in our design has been shown in Fig. 5. The function of the controller is to control the execution of different operations by sending appropriate control signals to different blocks in the execution unit. Execution of operations are achieved by moving data into the input register(s) of a functional block, triggering of the functional block, waiting for the completion of operation of the functional block and then moving the generated result from the output register of the functional block to the specified destination address in the register files. Sequence of operations performed by the controller is shown in fig.8 and their detailed description has been discussed below.

The controller used in the design performs a sequence of operations listed below in step1-step13. In each step different control signals are generated and a number of operations are performed.

1) **Step 1:** In this step, all the counters and accumulators are reset, all the addresses are initialized with their initial values and alpha_ACC’s are initialized with the bias values.

2) **Step 2:** In this step, input values are read from input_ROM (address is given by counter51) and

support vectors are read from all SV_BRAMs (address is given by counter663).

3) **Step 3:** In this step, values read in the previous step are given to the input registers of multiplier, 24 bits of input data is multiplied with the support vector and intermediate result of multiplication is stored in a register and controller goes to the next step.

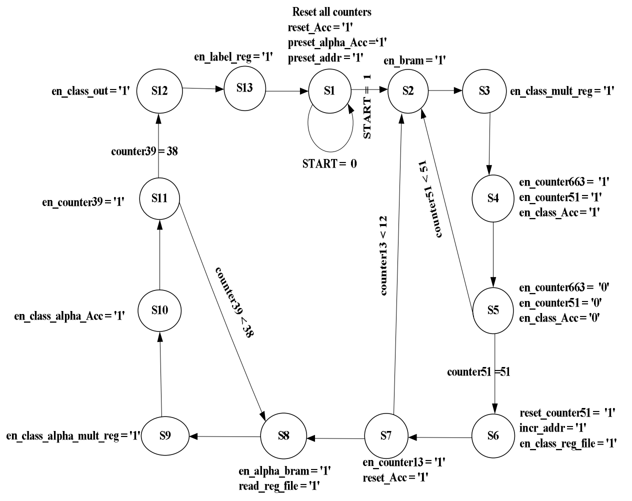


Fig. 8. Sequence of operations performed by the controller.

4) **Step 4:** In this step, output from multiplier computed in previous step are given to the first input registers of adder and values of all the accumulators (initially they contain zeros) are given to the second input registers of adder and controller goes to the next step. Counters counter51 and counter663 are incremented by 1 by giving their enable control signals.

5) **Step 5:** The counter counter51 is checked to see whether the counter51 count becomes 51 (1 row is completed or not). If the value of counter51 is less than 51, it again goes back to step 2. All the steps are repeated until the counter51 count becomes 51. After that it goes to the next step.

6) **Step 6:** In this step, all the accumulator's values computed in previous steps are stored in the register files. Addresses of all register files for all 3 classes are given by address0, address13 and address26 respectively.

7) **Step 7:** In this step, the counter counter13 is checked to see whether the counter13 count becomes 12 (13 rows are completed or not). If the value of counter13 is less than 12, it again goes back to step 2. All the steps are repeated until the counter13 count becomes 12. After that it goes to the next step.

8) **Step 8:** In this step, values are read from ALPHA_BRAMs (address is given by counter39) and all the accumulated values stored in register files (address is given by counter39).

9) **Step 9:** In this step, values read in the previous step are given to the input registers of multiplier blocks and controller goes to the next step.

10) **Step 10:** In this step, outputs from the multiplication computed in previous step are given to the first input registers of adders and values of all the ALPHA_ACC's (initially they contain bias values) are given to the second input registers of adders. Controller goes to the next step.

11) **Step 11:** In this step, the counter counter39 is checked to see whether the counter39 count becomes 38. If the value of counter39 is less than 38, it again goes back to step 9. All the steps are repeated until the counter39 count becomes 38. After that it goes to the next step.

12) **Step 12:** In this step, the storing of results computed from previous steps are done for all the 3 classes After that it goes to the next step.

13) **Step 13:** In this step, decision is made on the basis of maximum votes and stores result of the final classification by selecting the labels of different classes. Finally one of the labels among three labels (label1, label2 and label3 for class1, class2 and class3 respectively) will get enabled. After that it goes back to the initial state i.e. step 1.

IV. Simulation Result

All the modules of the proposed architecture are coded in VHDL and simulated using ModelSim 10.1C. Synthesis is carried out using Xilinx ISE tool chain (version 14.2). We have use 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 VIII shows comparison of FPGA simulation results of the proposed pairwise 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. The hardware resources used in the Virtes5 device for the implementation of the classifier is shown in table IX.

TABLE VIII. Comparison 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**	**9**	**0**	**0**	**9**
Error Rate		**2.3 %**			**2.3 %**	
Number of misclassified compared with MATLAB		**0**			**-**	
Maximum Frequency	**241.546 MHz, Virtex-IV**			**2.17 GHz, Intel Cor2Duo**		
Time of Computation		**4.140 ns**			**65 ms**	

TABLE IX. HARDWARE UTILIZATION RESULT OF THE PROPOSED ARCHITECTURE

Logic Utilization	Available	Used	Utilization Rate
Slice Flip Flop	81920	5148	6%
Slice LUTS	81920	6479	7%
Occupied Slice	20480	1980	9%
Bonded IOB	840	75	7%
BUFG	32	2	6%
RAMB36	298	4	1%
DSP48	320	24	7%

V. CONCLUSION

In this paper, the hardware architecture of pairwise linear Support Vector Machine for classification scheme, realized using VHDL and implemented on Xilinx ML510 FPGA platform, has been presented. The implemented architecture can perform real time classification operating at a frequency of 241.55 MHz. Classification accuracy of 97.87% has been achieved, which shows a good performance of our proposed architecture. The proposed architecture outperforms the reference architecture [18], in terms of speed and accuracy using much higher dimension feature size (51-d compared to 7-d of [24]).

ACKNOWLEDGMENT

The authors express their deep sense of gratitude to Dr. Chandra Shekhar, Director CSIR-CEERI for encouraging research and development activities. Authors, would like to thanks Dr. AS Mandal, Chief-Scientist and Prof. Raj Singh, Chief Scientist and Group Leader, IC Design Group, CSIR-CEERI, for their constructive suggestions. The authors would also like to thank Mr. Sanjeev Kumar, Technical Officer, IC Design Group, CSIR-CEERI, for providing technical support. The financial support of DeitY/MCIT is also gratefully acknowledged.

REFERENCES

[1] P. Ekman, "Emotion in the Human Face," Cambridge University Press, New York, 2nd edition, 1982.

[2] W. Hirata, J.K. Tan, H. Kim, et .al, "Recognizing facial expression for man-machine interaction," in Procedings of the IEEE ICCAS-SICE, 2009.

[3] M. Beszédeš, P. Culverhouse, and M. Oravecm,"Facial emotion classification using active appearance model and support vector machine classifier," Machine Graphics & Vision International Journal, vol. 18, no. 1, pp. 21-46, 2009.

[4] I. Kotsia, and I. Pitas, "Facial expression recognition in image sequences using geometric deformation features and support vector machines, " IEEE Transactions on Image Processing, vol. 16, no. 1, pp. 172-187, 2007.

[5] I. Kotsia, and I. Pitas, "Real time facial expression recognition from image sequences using support vector machines," in Visual Communications and Image Processing, International Society for Optics and Photonics, 2005.

[6] Dumas, Melanie, "Emotional expression recognition using support vector machines," in Proceedings of International Conference on Multimodal Interfaces, 2001.

[7] H.H. Tsai, Y.S Lai, and Y.C Zhang, "Using SVM to design facial expression recognition for shape and texture features," in Proceedings of the IEEE International Conference on Machine Learning and Cybernetics (ICMLC), vol. 5, 2010.

[8] R.A. Pati, V. Sahula, and A.S. Mandal, "Features classification using support vector machine for a facial expression recognition system," Journal of Electronic Imaging, vol. 21, no. 4, 2012.

[9] P. Visutsak, "Emotion Classification through Lower Facial Expressions using Adaptive Support Vector Machines," Journal of Man, Machine and Technology, vol. 2, no. 1, pp. 12-20, 2013.

[10] D. Anguita, A. Boni, and S. Ridella "A digital architecture for support vector machines: theory, algorithm, and FPGA implementation," IEEE Transaction on Neural Networks, vol. 14, no. 5, pp. 993-1009, 2003.

[11] F.M. Khan,M.G. Arnold, and W.M. Pottenger" Hardware-based support vector machine classification in logarithmic number systems," in Proceedings of the IEEE International Symposium on circuits and systems, vol. 5, pp. 5154-5157, 2005.

[12] K.M. Irick, M. DeBole, V. Narayanan, et al., "A hardware efficient support vector machine architecture for FPGA," in Proceedings of the 16th IEEE International Symposium on Field-Programmable Custom Computing Machines, pp. 304-305, 2008.

[13] C.F. Hsu, M.K. Ku, and L.Y. Liu," Support vector machine FPGA implementation for video shot boundary detection application," in Proceedings of the IEEE International Conference on SOC, pp. 239-242, 2009.

[14] S. Bauer, S. Kohler, K. Doll, et al.,"FPGA-GPU architecture for kernel SVM pedestrian detection," in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshop, pp. 61-68, 2010.

[15] O. Pina-Ramirez, R.Valdes-Cristerna, and O. Yanez-Suarez, "An FPGA implementation of linear kernel support vector machines," in Proceedings of the IEEE International Conference on Reconfigurable Computing and FPGA's, pp. 1-6, 2006.

[16] Z. Nie, X. Zhang, and Z.Yang, "An FPGA Implementation of Multi-Class Support Vector Machine Classifier Based on Posterior Probability," in Proceedings of 3rd International Conference on Computer and Electrical Engineering, pp. 296-302, 2012.

[17] D. Anguita, A. Ghio, S. Pischiutta, et al.," A Hardware-friendly Support Vector Machine for Embedded Automotive Applications," in Proceedings of the IEEE International Joint Conference on Neural Networks, 2007.

[18] D. Mahmoodi, A. Soleimani, H. Khosravi, et.al.,"FPGA simulation of linear and nonlinear support vector machine," Journal of Software Engineering and Applications, vol. 4, pp. 320-328, 2011.

[19] M. Ruiz-Llata, G. Guarnizo, and M. Yebenes-Calvino," FPGA implementation of a support vector machine for classification and regression," in Proceedings of the IEEE International Joint Conference on Neural Networks, pp. 1-5, 2010.

[20] M. Ruiz-Llata, and M. Yébenes-Calvino,"FPGA Implementation of support vector machines for 3D object identification," in Proceedings of the International Conference on Artificial Neural Networks, pp. 467-474, 2009.

[21] L. Shen, L. Bai, and M. Fairhurst," Gabor wavelets and general discriminant analysis for face identification and verification," Image and Vision Computing, vol. 25, no. 5, pp. 553-563, 2007.

[22] M.B.A. Haghighat, and E. Namjoo,"Evaluating the informativity of features in dimensionality reduction methods," in Proceedings of the 5th IEEE International Conference on Application of Information and Communication Technologies, pp. 1–5, 2011.

[23] L. Shen, and L. Bai,"AdaBoost Gabor feature selection for classification," in Proceedings of Image and Vision Computing, New Zealand, pp. 77-83, 2004.