Journal of Software Engineering and Applications, 2011, 4, 320-328
doi:10.4236/jsea.2011.45036 Published Online May 2011 (http://www.SciRP.org/journal/jsea)

FPGA Simulation of Linear and Nonlinear Support Vector Machine

Davood Mahmoodi[1], Ali Soleimani[1], Hossein Khosravi[1], Mehdi Taghizadeh[2]

1Department of Electrical Engineering and Robotic, Shahrood University of Technology, Shahrood, Iran; 2Department of Electrical Engineering, Kazerun Branch, Islamic Azad University, Kazerun, Iran
Email: davood.mahmoodi@gmail.com, sorlimani_ali@shahroodut.ac.ir, hosseinkhosravi@gmail.com, mehdi.taghizade@gmail.com

Received April 10th, 2011; revised April 30th, 2011; accepted May 8th, 2011.

ABSTRACT

Simple hardware architecture for implementation of pairwise Support Vector Machine (*SVM*) *classifiers on FPGA is presented. Training phase of the SVM is performed offline*, *and the extracted parameters used to implement testing phase of the SVM on the hardware. In the architecture*, *vector multiplication operation and classification of pairwise classifiers is designed in parallel and simultaneously. In order to realization*, *a dataset of Persian handwritten digits in three different classes is used for training and testing of SVM. Graphically simulator*, *System Generator*, *has been used to simulate the desired hardware design. Implementation of linear and nonlinear SVM classifier using simple blocks and functions*, *no limitation in the number of samples*, *generalized to multiple simultaneous pairwise classifiers*, *no complexity in hardware design*, *and simplicity of blocks and functions used in the design are view of the obvious characteristics of this research. According to simulation results*, *maximum frequency of* 202.840 *MHz in linear classification*, *and classification accuracy of* 98.67% *in nonlinear one has been achieved*, *which shows outstanding performance of the hardware designed architecture.*

Keywords: *Hardware Architecture*, *Support Vector Machine*, *FPGA*, *Persian Handwritten Digits*, *System Generator*

1. Introduction

Today, many algorithms in the fields of signal processing, including speech processing, image processing, machine vision, data mining and pattern recognition, is developed on software and every day new progress in their development is achieved. Many of these algorithms run with lower frame rates because of computational complexity. Fast and real-time processing requirement yield hardware implementation of these algorithms. But, implementing these algorithms on hardware such as microprocessors, DSP or FPGA processors continue to be an important challenge [1].

Recently FPGAs have been introduced as one of the hardware used in the field of digital signal processing and provided acceptable results in real-time applications, using parallel processing [2].

SVM is one of the trusted and ultra high-performance algorithms, has been widely used in the fields of linear and nonlinear classification and regression problems [3]. As regards that the structure of SVM includes a second optimization process, so practical applications of this classifier will be limited, due to the computational complexity and power consumption of training phase.

Obviously useful applications of online SVM require a hardware implementation suitable for training and testing phase. But considering the type of application, training phase of SVM can be performed offline in a computer by software. Therefore in these cases only testing phase of the SVM, that includes a decision function for classification target and is performed very fast, will be implemented on hardware [4].

Anguita [5] proposed a fully digital hardware structure for implementation of testing and training phase of SVM using linear and RBF kernel functions. The proposed structure in addition to consuming too much hardware resources, has not an acceptable processing speed. Maximum frequency obtained by Anguita was 35.3 MHz. Faisal Khan [6] avoided of fixed-point computations and used the logarithmic number system for implementing testing phase of linear SVM. Although the proposed structure is suitable for hardware, but always may be difficulties in converting real numbers to their equivalent logarithmic. Ramirez [4] implemented a linear SVM for classification of three-dimensional MRI images. Classification accuracy and processing time was 95% and 109.7s respectively, which was 5% less and 1.84 times more than a 550 MHz PC. In this research maximum frequency of 202.840 MHz in linear classification, and classification accuracy of 98.67% in nonlinear one has been achieved, while processing time is much more than PC, which is comparable in literatures [1,7,8].

The rest of this paper is organized as follow: after introduction, Section 2 describes the basics of linear, nonlinear and multiclass SVM. In Section 3 proposed algorithm for SVM hardware implementation, include-ing software implementation and hardware architecture design has been described. Finally our simulation results are given in Section 4, followed by conclusion and suggestions in Section 5.

2. Support Vector Machine

SVM is a technique of classification and regression introduced in 1990s by Vapnik [9,10]. SVM is essentially a binary classification, but possible to classify samples of multiple classes. Also it can be used to solve either linear and nonlinear classification or regression problems.

2.1. Linear SVM

Consider a linear classification problem aiming to find optimal separating hyperplane with maximum margin. Suppose $\boldsymbol{x_i}$ is the feature vector set of training samples that are linearly separable and have been labeled in classes1 and 2, as **Figure 1**. There might be some data that fall into the margin, called slack variables.

So decision function is defined as follow:

$$d(\boldsymbol{x}) = sign\left(\boldsymbol{w}^T \boldsymbol{x} + b\right) \tag{1}$$

Optimal separating hyperplane can be obtained by solving the following dual Lagrange problem:

$$\text{Maximize} \quad L_d(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} y_i y_j \alpha_i \alpha_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

$$\text{Subject to} \quad \begin{cases} 0 \le \alpha_i \le C \\ \sum_{i=1}^{N} \alpha_i y_i = 0 \end{cases} \tag{2}$$

in which C is a trade-off parameter between maximization the margin and minimization the error, and is determined by user [11].

By solving dual Lagrange function, $\boldsymbol{\alpha}$ is obtained. Subsequently $\boldsymbol{w}$ and b are achieved too, according to the following:

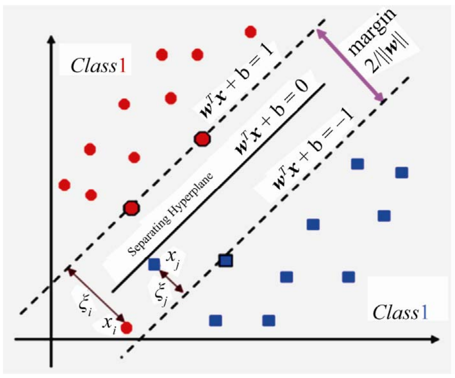

Figure 1. Optimal separating hyperplane with maximum margin and slack variables.

$$\boldsymbol{w} = \sum_{i=1}^{SV} \alpha_i x_i y_i \tag{3}$$

$$b = y_i - \boldsymbol{w}^T \boldsymbol{x}_i \tag{4}$$

where SV is the number of support vectors.

Now we can classify an unknown data $\boldsymbol{x}$, by utilizing (3) and (4) into the decision function Equation (1), which yield to the following equation:

$$d(\boldsymbol{x}) = sign\left(\sum_{i=1}^{SV} \alpha_i y_i \boldsymbol{x}_i^T \boldsymbol{x} + b\right) \tag{5}$$

2.2. Nonlinear SVM

A question that is being asked here is that what to do if data are not linearly separable? Is it possible to generalize the idea of a linear SVM to nonlinear systems? Extensive studies conducted in this area resulted in Mercer theory [12]. The idea behind this theory is to transfer vector $\boldsymbol{x}$ from limited space (input space) to the higher space (feature space), using Hilbert conversion, as **Figure 2**. In this context a vector $\boldsymbol{x}$ in the feature space is converted into the $\varphi(\boldsymbol{x})$.

According to this theorem, decision function of Nonlinear SVM can be acquired as following:

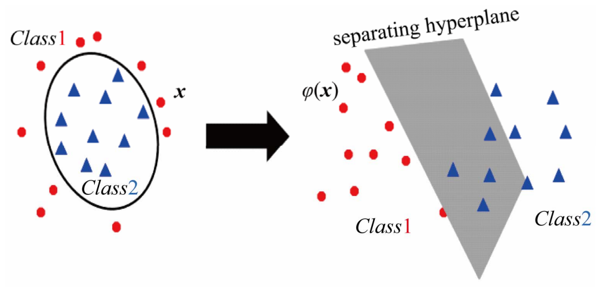

Figure 2. Transferring a data *x* form input space (left) into the feature space (right).

$$d(\boldsymbol{x}) = sign\left(\sum_{i=1}^{SV} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b\right) \quad (6)$$

where:

$$b = y_i - \sum_{i=1}^{SV} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) \quad (7)$$

in which the term $\varphi(\boldsymbol{x}_i)^T \varphi(\boldsymbol{x})$ has been replaced with a kernel function defined as follow:

$$K(\boldsymbol{x}_i, \boldsymbol{x}) = \varphi(\boldsymbol{x}_i)^T \varphi(\boldsymbol{x}) \quad (8)$$

where, well-known kernel functions have been introduced in literatures [13-15].

2.3. Multi Class SVM

SVM is a binary classifier, but some methods can be used to solve multi-class problems with this classifier. Four methods of multi-class SVM are:

- One against All Method
- Pairwise Classifiers Method
- Error-correcting Output Code (ECOC) Method
- All Classes at Once Method

In one against all method, if we have *n* class, *n* binary SVM classifier will be formed, so that in *i*-th classifier, class *i* is separated from the others. But in this status still there will be unclassified regions according to **Figure 3** [11].

To solve this problem, *Krebel* [16] proposed pairwise classifiers method, in which to solve *n* class problem, $n(n-1)/2$ binary classifier will be formed. This method compared to one against all is better but still there will be unclassified region according to **Figure 4**.

In the pairwise classifier method, for classification of an input data $\boldsymbol{x}$, in each binary decision function $d_{ij}(\boldsymbol{x})$ a class is selected, and finally the one with the most votes is selected as desired class.

3. The Proposed Algorithm for SVM Hardware Implementation

To implement SVM classifier architecture on the hardware, we must first perform learning phase of the SVM, which has a complicated algorithm, offline on the software, and then use extracted parameters to implement testing phase on the hardware. In this research training phase of the SVM is implemented in the MATLAB software environment utilizing libsvm [17] model, and the graphical hardware simulator, System Generator, will be used to design and implement of testing phase.

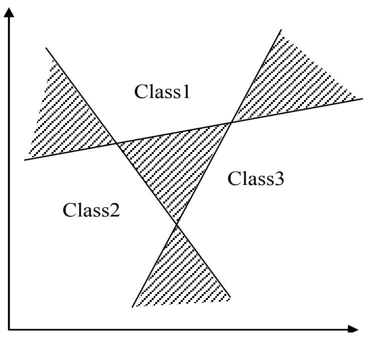

Figure 3. Unclassified regions in one against all method.

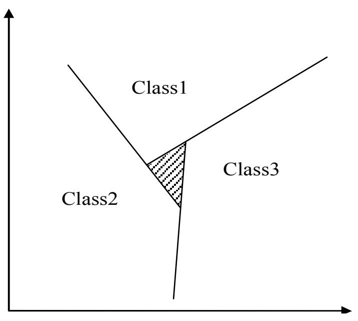

Figure 4. Unclassified region in pairwise classifiers method.

In order to realize designed architecture and desired results, three different classes of data related to Persian handwritten digits, 1, 4 and 8, are used for training and testing samples of the SVM [18]. We named class1 for 1, class2 for 4, and class3 for 8. So the architecture will be designed for a 3-class SVM, according to pairwise method. 800 data from each class (total 2400 samples) is provided. From each class of data, 600 samples have been used for training and 200 samples for testing. Data has feature vector dimension of 7 and 24, which have been extracted by using of geometric moments approach [19]. We use 7-*d* data for linear and 24-*d* data for nonlinear SVM.

3.1. Software Implementation of SVM

Before hardware implementing of the SVM architecture, it is necessary to run both training and testing phase in MATLAB. The target is to get the lowest classification error rate by changing various parameters. In additions, in that lowest error rate, necessary parameters will be extracted for offline classification on the hardware. In order to implement software SVM, despite SVM toolbox of the MATLAB, many other codes written by researchers have been proposed. Libsvm model has been introduced as one of the best and trusted tools for implementation of SVM, and is used in most of academic papers and researches, so we utilize Libsvm too.

In general, before SVM implementation, the type of kernel function must be specified. We used linear and Gaussian kernel functions to implement SVM.

3.2. SVM Software Implementation with Linear Kernel

Linear kernel is the simplest type of kernel functions.

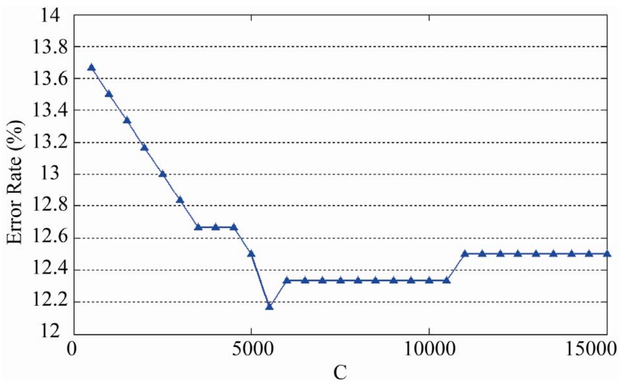

Figure 5. Diagram curve of linear SVM classification error rate versus changing parameter C.

Because of its simplicity in structure, and fast computation, it is a very good choice to for hardware implementation. In this research, experiments show that by changing the parameter C, the classification error rate will be in change too. In order to achieve the lowest error rate, we draw diagram curve of error rate versus changing parameter C of SVM structure. **Figure 5** shows this diagram in classification of Persian handwritten digits with *7-d* data.

Figure 5 shows that the best classification error rate is obtained at $C = 5500$. But if the same value of C is used for implementation on hardware, have to assign much number of bits. If the value of C is equal to 100, due to the low number of bits, desired hardware design can be implemented, while increasing in error rate is not very sensible.

Table 1 shows numerical results of linear SVM classification by using $C = 100$, in which *SV* and b indicate the number of support vectors and bias term, respectively.

3.3. SVM Software Implementation with Nonlinear Kernel

Gaussian kernel function is one of the kernel functions that mostly has been used in support vector machines, and provided much better results compared to other ones [20]. In this research, the Gaussian kernel function is used for training and testing nonlinear SVM. Here the results show that by changing the parameter C, classification error rate will not be changed anymore. But instead, there is a parameter γ in Gaussian kernel, affecting the classification error rate, which defined as following:

$$\gamma = 1/2\sigma^2 \tag{9}$$

Choosing appropriate *gamma* value is a very important issue to find the optimal classification error rate. Exact formulation doesn't exist to calculate an appropriate value of *gamma*, and often try and error method is used to get that. In order to attain that value in nonlinear classification of Persian handwritten digits with 24-*d* samples, a diagram curve of error rate versus changing of *gamma* is considered in **Figure 6**, which shows that *gamma* = 0.95 offers the lowest error rate. So in the next step we use *gamma* = 0.95 and again run training and testing phases of SVM. **Table 2** shows numerical results of this classification.

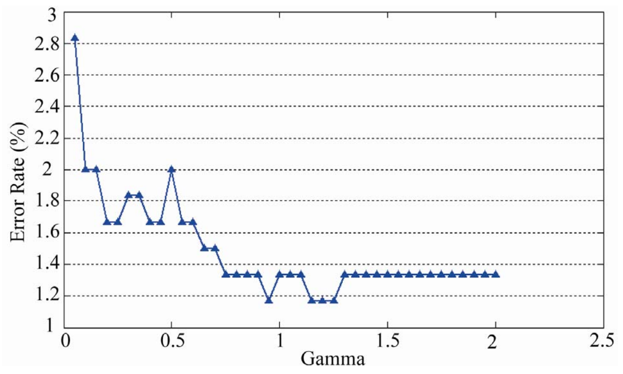

Figure 6. Diagram curve of nonlinear classification error rate versus changing parameter γ.

3.4. Hardware Architecture Design

Figure 7 shows block diagram of hardware architecture design for 3-class SVM classification according to simultaneous pairwise classifiers method. This design is the same for linear and nonlinear SVM, but only *kernel* block is differently designed.

3.5. System Generator Design of Linear SVM

Before FPGA implementation, in order to simplify computational operations it is better to merge dot product of vectors $\boldsymbol{\alpha}$ and $\boldsymbol{y}$ into a new vector called $\boldsymbol{y\alpha}$. **Figure 8** shows matrix form of linear SVM decision function for classification of *Class*1 & 2, *i.e.* $d_{12}(\boldsymbol{x})$.

Table 1. Numerical results of linear SVM classification with C=100.

	Class1 & 2		Class1 & 3		Class2 & 3	
	Class1	Class2	Class1	Class2	Class1	Class2
SV	35	35	6	7	243	243
b	−0.7895		3.7228		−10.4076	
Error Rate	10.25%		10.5%		20.75%	

Table 2. Numerical results of nonlinear SVM classification with γ = 0.95.

	Class1 & 2		Class1 & 3		Class2 & 3	
	Class1	Class2	Class1	Class2	Class1	Class2
SV	19	22	18	18	33	35
b	−3.8358		−3.2601		−3.1587	
Error Rate	1.5%		0.5%		1.5%	

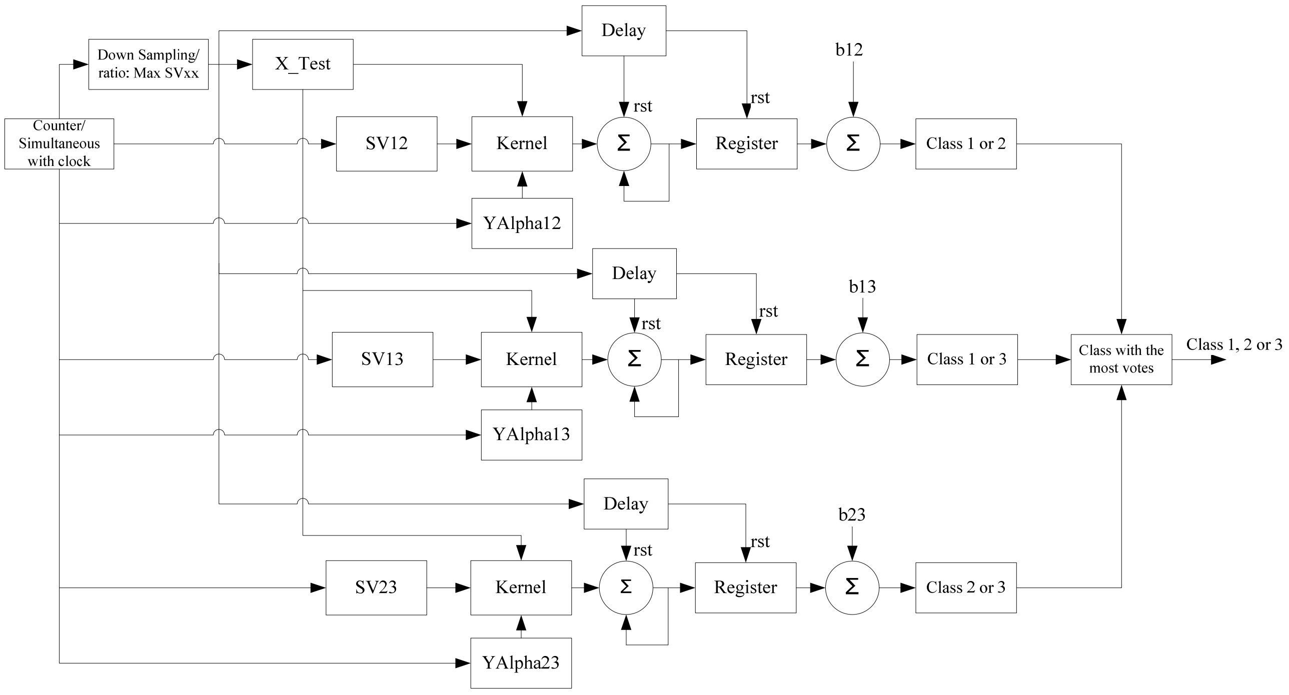

Figure 7. Block diagram of 3-class SVM hardware architecture design.

To design hardware architecture of above function, a combination of series and parallel methods have been used. **Figure 9** shows part of designed architecture using System Generator. Each array of rows of matrix *SV* is located in *SV ROM* 1 to *SV ROM* 7 outputs, which called by counter1. Test data are in blocks *X_Test ROM*1 to *X_Test ROM* 7 addressing by *Counter* 2 with a period of 70 (as the number of support vectors of *Class*1 & 2).

Vector of ***yα*** located in *YAlpha ROM* which its counter is the same as *SV ROM*s. Blocks of *Mult*1 to *Mult*7 and *Addsub*1 to *Addsub*6 perform inner product between test data and support vectors; this result is multiplied by corresponding array of ***yα*** vector in *Mult*9. The remaining operations is sum of this values, that is done serially by *Accumulator* and +*b* blocks.

3.6. System Generator Design of Nonlinear SVM

For implementation of nonlinear SVM, 24-*d* data from Persian handwritten digit database has been used. Decision function equation of Nonlinear SVM with Gaussian kernel function is as follows:

$$d(\boldsymbol{x}) = \sum_{i=1}^{SV} \alpha_i y_i \exp(-\gamma \|x - x_i\|^2) + b \qquad (10)$$

Due to the structural limitations in the FPGA, it is better to simplify above function, before implementation of it. Suppose that ***A*** is a vector as follows:

$$\boldsymbol{A} = \begin{bmatrix} a_1 & a_2 & \cdots \end{bmatrix} \qquad (11)$$

Norm A is defined as:

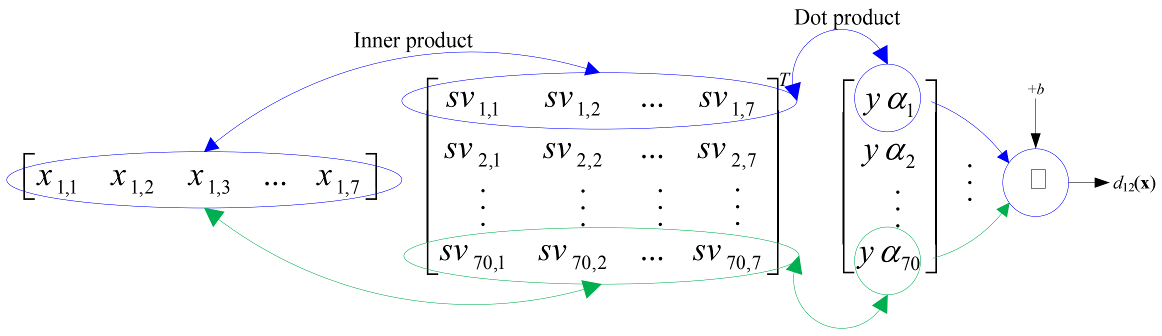

Figure 8. Matrix form of linear SVM decision function.

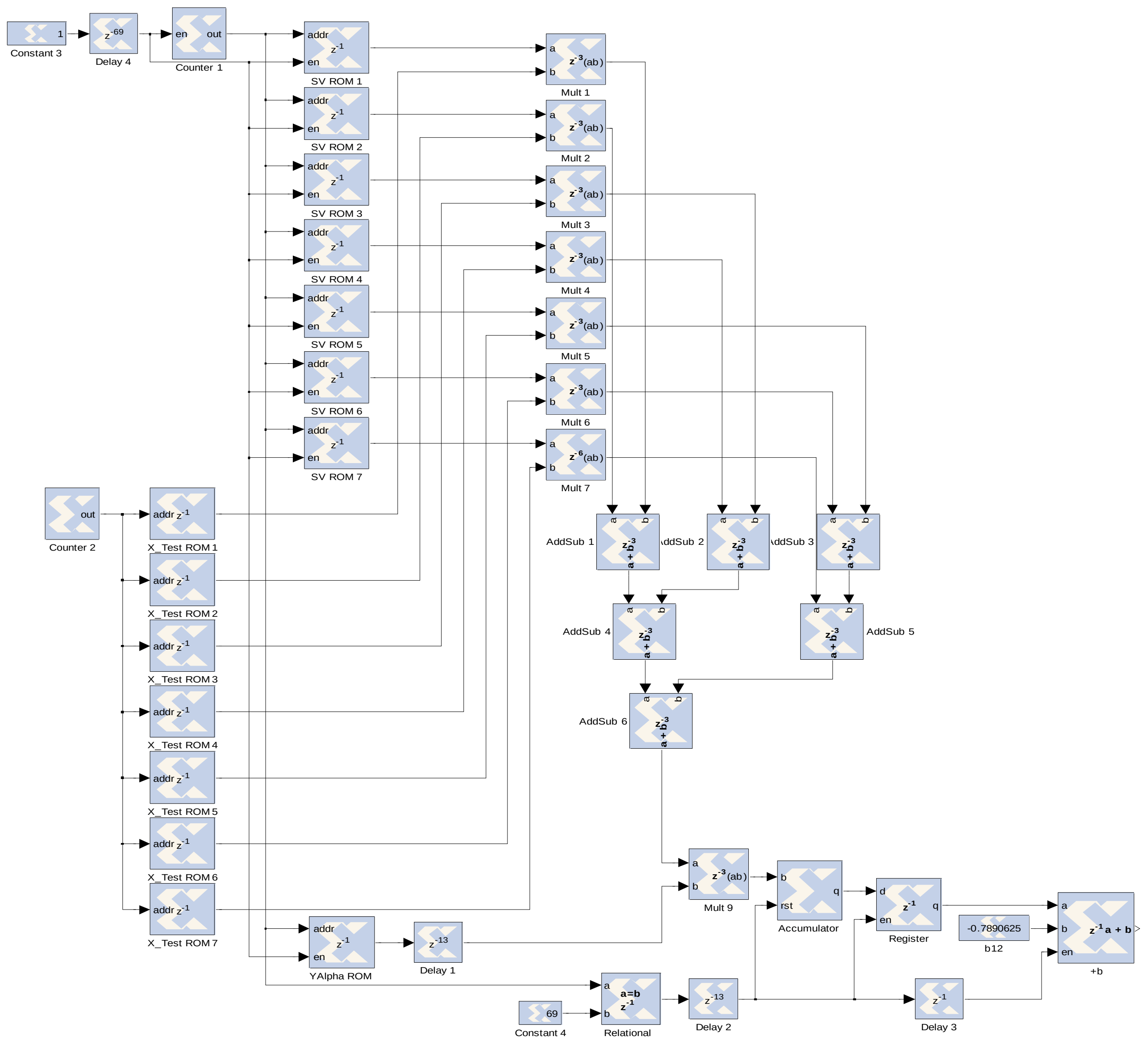

Figure 9. Hardware architecture for computation of $\alpha_i y_i x^T x_i$.

$$\|A\| = \left(|a_1|^2 + |a_2|^2 + \ldots\right)^{1/2} \qquad (12)$$

And Square of norm A:

$$\|A\|^2 = \left(|a_1|^2 + |a_2|^2 + \ldots\right) \qquad (13)$$

The above equation can be simpler as follows:

$$\|A\|^2 = \left(a_1^{\ 2} + a_2^{\ 2} + \ldots\right) \qquad (14)$$

So we can easily implement $\|x - x_i\|^2$ by using (14). For implementation of *exp* function, there is a *CORDIC* block in System Generator that produces *sinh* and *cosh* outputs. By knowing following equation, *exp* function can be implemented utilizing a *CORDIC* block and an *Adder*.

$$exp(x) = Sinh(x) + Cosh(x) \qquad (15)$$

Figure 10 shows FPGA architecture for computation of $\alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x})$ in nonlinear Gaussian SVM, while other parts of design are the same as linear one, except that here data dimension is 24-*d*. The value of γ is located in *constant*2 block.

4. Simulation Results

First step for simulation of designed architecture is to choose FPGA part number. We use Xilinx Virtex4-xc4vsx35 device for linear and nonlinear SVM simula-

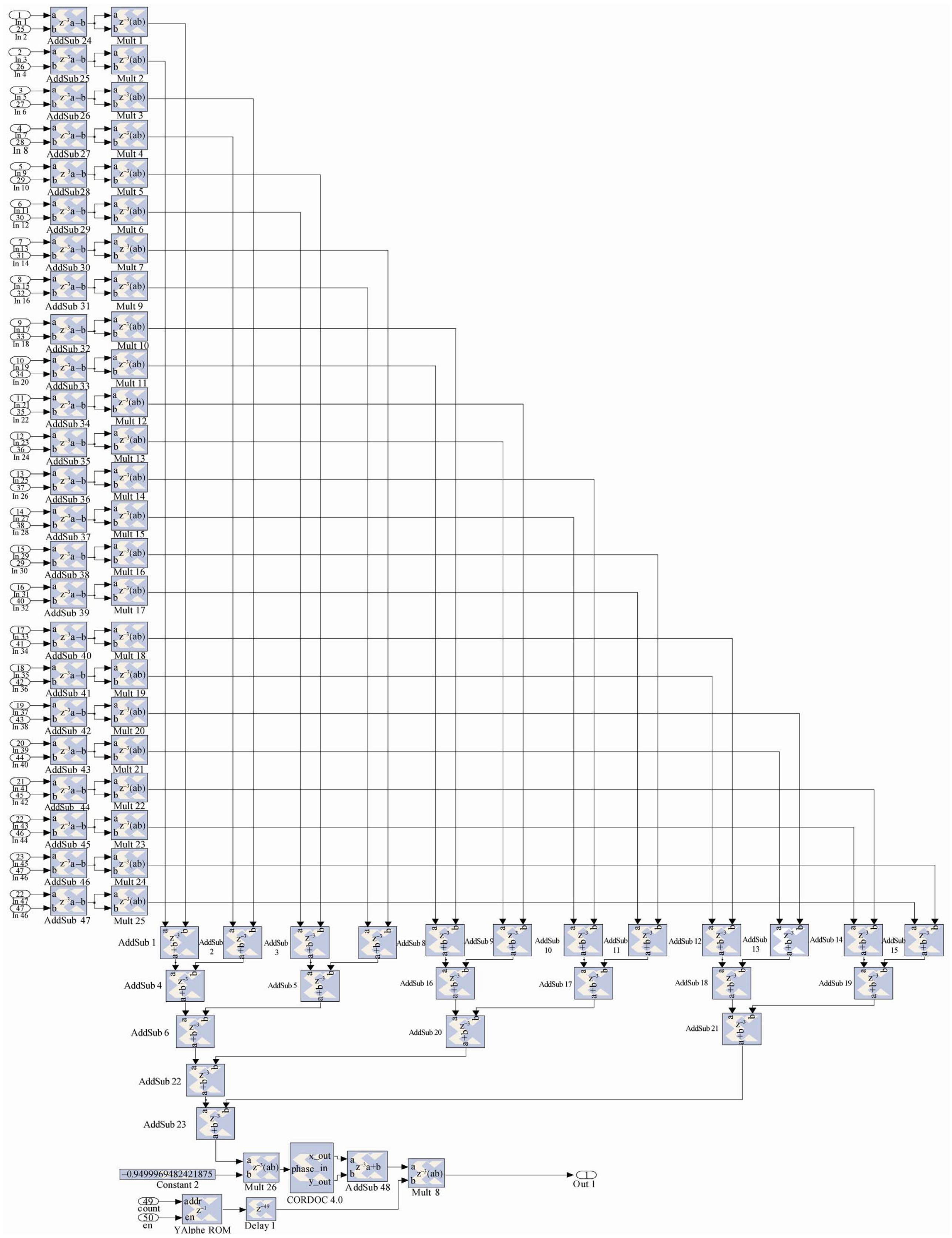

Figure 10. Hardware architecture for computation of αiyi K(*x*, *xi*).

Table 3. Hardware simulation results of linear SVM compared with MATLAB.

	System Generator			MATLAB		
	Class1	Class2	Class3	Class1	Class2	Class3
*Class*1	200	0	0	200	0	0
*Class*2	6	169	25	6	159	35
*Class*3	0	41	159	0	42	158
Error Rate		12%			13.83%	
Number of misclassified compared with MATLAB		11			–	
Maximum Frequency	202.840 MHz, Virtex-IV			2.1 GHz, Intel Cor2Dou		
Time of computation	1.4 ms			0.45 s		

tion. Fixed point numbers is used for quantization. For linear simulation Q24.16 and for nonlinear one Q24.14 quantization format is used. To avoid increasing the word length in serial computations, truncation method is used.

4.1. Linear SVM Simulation

Table 3 shows FPGA simulation results of linear SVM hardware architecture compared with software implementation in MATLAB. Rows of 1 through 3 show the number of samples of each class that classify into the correct class or misclassify into the others.

It might seem slightly strange that error rate in System Generator simulation is lower than MATLAB, but we know that because of quantization errors in System Generator maybe some test samples classify into correct class, while misclassified in MATLAB.

The considerable result of this table is maximum frequency and total time of computation of classification which are 202.840 MHz and 1.4 ms, respectively.

In **Table 4** hardware resources used in Virtex4 device for implementation of above classifier is shown.

Table 4. Hardware resources used in Virtex4 device for implementation of linear SVM.

Logic Utilization	Available	Used	Utilization Rate
Slice Flip Flop	30 720	1422	4%
4 *input LUT*	30 720	748	2%
Occupied Slice	15 360	849	5%
Bonded IOB	448	167	37%
BUFG	32	1	3%
*RAMB*16	192	31	16%
*DSP*48	192	27	14%

Table 5. Hardware simulation results of nonlinear SVM compared with MATLAB.

	System Generator			MATLAB		
	Class1	Class2	Class3	Class1	Class2	Class3
*Class*1	199	1	0	199	1	0
*Class*2	3	194	3	3	195	2
*Class*3	0	1	199	0	1	199
Error Rate		1.33%			1.17%	
Number of Misclassified Compared with MATLAB		1			–	
Maximum Frequency	151.286 MHz, Virtex-IV			2.1 GHz, Intel Cor2Dou		
Time of Computation	0.27 ms			0.11 s		

4.2. Nonlinear SVM Simulation

Simulation result of hardware designed architecture of nonlinear SVM is shown in **Table 5**. In this case classification error rate of 1.33% and total time of computation of 0.27 ms are considerable.

In **Table 6** hardware resources used in Virtex4 device for implementation of nonlinear SVM classifier is shown.

5. Conclusions

In this research FPGA architecture of linear and nonlinear pairwise SVM classifiers for detection of 3-class of Persian handwritten digits has been proposed. This design can be generalized to other SVM classification applications with no limitation in the number of training and testing data. Also it is Possible to increase the number of pairwise classifiers. But there is a restriction only on the FPGA hardware resources. This problem can be solved by utilizing multi FPGAs or some external memory devices.

In order to continue the research, proposing a method for implementation of other nonlinear kernel functions is recommended. Since that for a certain application one of

Table 6. Hardware resources used in Virtex4 device for implementation of nonlinear SVM.

Logic Utilization	Available	Used	Utilization Rate
Slice Flip Flop	30720	11589	37%
4 *input LUT*	30720	9141	29%
Occupied Slice	15360	7261	47%
Bonded IOB	448	241	53%
BUFG	32	1	3%
*RAMB*16	192	99	51%
*DSP*48	192	81	42%

the kernel functions respond better classification results than the others, if the solutions to implement all these functions exist, the designer will have a tendency to achieve the best classification accuracy.

Another suggestion is to implement the entire process of SVM, including training and testing phase, on the FPGA. Training phase of the SVM includes optimization problem Equation (2), which has a very time consuming and complicated process of solution for hardware implementation purpose. If an appropriate and hardware friendly solution is found to simplify the problem, then whole process of SVM implemented on the FPGA, so that no need to implement the training phase in software environment which may cause quantization errors in testing phase hardware implementation.

REFERENCES

[1] M. Ruiz-Llata and M. Yébenes-Calvino, "FPGA Implementation of Support Vector Machines for 3D Object Identification," *Artificial Neural Networks–ICANN* 2009, Limassol, 14-17 September, 2009, pp. 467-474.

[2] U. Meyer-Baese, "Digital Signal Processing with Field Programmable Gate Arrays," Springer Verlag, Berlin, 2007.

[3] A. Ganapathiraju, *et al.*, "Applications of Support Vector Machines to Speech Recognition," *IEEE Transactions on Signal Processing*, Vol. 52, No. 8, 2004, pp. 2348-2355. doi:10.1109/TSP.2004.831018

[4] O. Pina-Ramfrez, *et al.*, "An FPGA Implementation of Linear Kernel Support Vector Machines," *IEEE International Conference on the Reconfigurable Computing and FPGA's*, San Luis, September 2006, pp. 1-6.

[5] D. Anguita, *et al.*, "A Digital Architecture for Support Vector Machines: Theory, Algorithm, and FPGA Implementation," *Neural Networks*, *IEEE Transactions on*, Vol. 14, No. 5, 2003, pp. 993-1009. doi:10.1109/TNN.2003.816033

[6] F. M. Khan, *et al.*, "Hardware-Based Support Vector Machine Classification in Logarithmic Number Systems," *Circuits and Systems*, Vol. 5, 2004, pp. 5154-5157.

[7] C. F. Hsu, *et al.*, "Support Vector Machine FPGA Implementation for Video Shot Boundary Detection Application," *System-on-Chip Conference*, 4-5 November 2009, pp. 239-242.

[8] M. Moradi, *et al.*, "A New Method of FPGA Implementation of Farsi Handwritten Digit Recognition," *European Journal of Scientific Research*, Vol. 39, No. 3, 2010, pp. 309-315.

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

[10] V. Vapnik, "The Nature of Statistical Learning Theory," Springer Verlag, Berlin, 1995.

[11] S. Abe, "Support Vector Machines for Pattern Classification (Advances in Pattern Recognition)," Springer-Verlag New York, 2005.

[12] M. Martínez-Ramón and C. G. Christodoulou, "Support Vector Machines for Antenna Array Processing and Electromagnetics," Morgan & Claypool, California, 2006.

[13] J. Diederich, "Rule Extraction from Support Vector Machines," Springer-Verlag, New York, 2008. doi:10.1007/978-3-540-75390-2

[14] B. Schölkopf and A. J. Smola, "Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond," The MIT Press, Cambridge, 2002.

[15] T. M. Huang, *et al.*, "Kernel Based Algorithms for Mining Huge Data Sets: Supervised, Semi-Supervised, and Unsupervised Learning," Springer Verlag, Berlin, 2006.

[16] U. H. G. Kreßel, "Pairwise Classification and Support Vector Machines," MIT Press, Cambridge, 1999.

[17] C. Chih-Chung and L. Chih-Jen, "LIBSVM: A Library for Support Vector Machines," 2001. http://www.csie.ntu.edu.tw/~cjlin/libsvm

[18] H. Khosravi and E. Kabir, "Introducing a Very Large Dataset of Handwritten Farsi Digits and a Study on Their Varieties," *Pattern Recognition Letters*, Vol. 28, 2007, pp. 1133-1141. doi:10.1016/j.patrec.2006.12.022

[19] S. Theodoridis and K. Koutroumbas, "Pattern Recognition," 3rd Edition, Elsevier, Amsterdam, 2006.

[20] H. T. Lin and C. J. Lin, "A Study on Sigmoid Kernels for SVM and the Training of Non-PSD Kernels by SMO-Type Methods," Department of Computer Science and Information Engineering, National Taiwan University, Taiwan, 2003.