Volume 47, Number 1, 2006 **ACTA TECHNICA NAPOCENSIS**
Electronics and Telecommunications

A PARTIALLY SEQUENTIAL-PARTIALLY PARALLEL ANALOG IMPLEMENTATION OF A SVM CLASSIFIER

Gabriel OLTEAN Mihaela GORDAN Sorin HINTEA
Technical University of Cluj-Napoca,
Str. C. Daicoviciu 15, 400020, Phone 026440147, Gabriel.Oltean@bel.utcluj.ro

Abstract: The classification task in large data spaces can now relay on newly developed Support Vector Machine technique. A trade-off between the hardware complexity and processing time is a partially sequential – partially parallel implementation of the processing issues based on a description of the vectors as time-continuous signals. We present an analog implementation of a SVM classifier that uses an analog multiplier to compute the dot product between each support vector and the test vector, in a sequential manner. All the dot products necessary to classify a test vector are implemented in parallel. The correct operation and the accuracy of our analog SVM classifier is proven on the IRIS dataset.

Key words: *SVM classifier; analog multiplier, analog signal processing.*
.

A PARTIALLY SEQUENTIAL-PARTIALLY PARALLEL ANALOG IMPLEMENTATION OF A SVM CLASSIFIER

Gabriel OLTEAN Mihaela GORDAN Sorin HINTEA
Technical University of Cluj-Napoca,
Str. C. Daicoviciu 15, 400020, Phone 026440147, Gabriel.Oltean@bel.utcluj.ro

Abstract: The classification task in large data spaces can now relay on newly developed Support Vector Machine technique. A trade-off between the hardware complexity and processing time is a partially sequential – partially parallel implementation of the processing issues based on a description of the vectors as time-continuous signals. We present an analog implementation of a SVM classifier that uses an analog multiplier to compute the dot product between each support vector and the test vector, in a sequential manner. All the dot products necessary to classify a test vector are implemented in parallel. The correct operation and the accuracy of our analog SVM classifier is proven on the IRIS dataset.

Key words: *SVM classifier; analog multiplier, analog signal processing.*

I. INTRODUCTION

In its basic form, a SVM is a binary classifier based on the optimal separating hyperplane algorithm that implements the Structural Risk Minimisation principle. Based on a training set the SVM learning algorithm derives the so-called optimal separating hyperplane, which will perfectly separate the positive and negative examples, with maximal distance to the hyperplane [1]. The training vectors x_s, $x_s \in R^N$, s=1,2,...,N_s, that are the closest positive and closest negative to the hyperplane are called the support vectors of the SVM. Their labels y_s (+1 for a positive example and -1 for the negative example) and their associated positive Lagrange multipliers α_s, define the decision function of the SVM classifier as:

$$f(x) = \sum_{s=1}^{Ns} \alpha_s y_s K(x, x_s) + b \qquad (1)$$

where:
x denotes a test vector to be classified by the SVM, $x \in R^N$;
b represents the bias term of the hyperplane;
$K(\cdot,\cdot)$ represents a kernel that computes the dot product of $\boldsymbol{x}$ and $\boldsymbol{x}_s$ either in their original space R^N (for a linear SVM) or in a higher dimensional feature space R^M, $M>>N$ (for non-linear SVMs) [1].

The number of the support vectors N_s increases with the complexity of the classification task. Furthermore, difficult classification problems are in general associated to a large dimensionality of the vectors space, N – large. This increases the computational complexity of the SVM classification.

A great effort was devoted to find computationally efficient implementations. Reducing the numerical complexity of the classification phase is extremely important for real-time applications, as e.g. video-sequence analysis.

Both hardware and software solutions are reported in the literature; the software solutions, strictly sequential, are based on reducing the dimension of the feature space N and the number of support vectors N_s. In the hardware implementations, the duration of the classification phase is reduced by parallelising to the largest possible extent the computation of $f(\boldsymbol{x})$. Analog implementations recently reported in the literature [2] make use of the possibility to perform in parallel the kernel evaluations and the N feature-by-feature multiplies in $\boldsymbol{x}^T\boldsymbol{x}_s$. This leads to massively parallel circuit structures, difficult to implement for large problems.

A compromise solution, partially sequential – partially parallel, could provide a good trade-off between the duration of the classification phase and the hardware complexity. The proposed approach is the implementation of each dot product $\boldsymbol{x}^T\boldsymbol{x}_s$, s=1,2,...,N_s, in a sequential analog fashion, whereas the N_s dot product computations (kernel evaluations) will be done in parallel. A number of N_s parallel analog processing channels are needed to evaluate $f(\boldsymbol{x})$. The computation of $\boldsymbol{x}^T\boldsymbol{x}_s$ in the analog sequential way is achieved by providing time-continuous descriptions $x(t)$ and $x_s(t)$ of the test vector $\boldsymbol{x}$ and support vectors $\boldsymbol{x}_s$. Thus the computation $\boldsymbol{x}^T\boldsymbol{x}_s$ can be formulated as an analog multiplication followed by integration, which can be implemented with simple analog structures.

The purpose of this paper is to build an analog circuit with analog multipliers and op amps able to perform the SVM classification in such a partially sequential, partially parallel fashion. The circuit is developed for a classifier with two support vectors and simulated in the Orcad CAD tool.

II. TIME-CONTINUOUS CLASSIFICATION PHASE

For the time being, we will consider in equation (1) the use of the linear kernel, $K(\boldsymbol{x},\boldsymbol{x}_s)=\boldsymbol{x}^T\boldsymbol{x}_s$. According to our implementation goal, we need to express $\boldsymbol{x}^T\boldsymbol{x}_s$ in a time-continuous fashion.

Considering $\mathbf{x}$ and $\boldsymbol{x}_s$ as sampled versions of two signals $x(t)$ and $x_s(t)$, $\boldsymbol{x}^T\boldsymbol{x}_s$ is expressed as:

$$x^T x_s = \int_{t=0}^{N/f_S} x(t)x_s(t)dt \qquad (2)$$

which gives the basic of the time-continuous formulation of the SVM classification proposed; f_s represents the sampling frequency [3].

With this formulation, the SVM classification phase can be implemented as follows:

- A sequential computation of each dot product $\boldsymbol{x}^T\boldsymbol{x}_s$, s=1,2,...,$N_s$ using an analog multiplier block with the input signals $x(t)$ and $x_s(t)$, followed by an integrator whose output is read at the moment N/f_S and multiplied by $\alpha_s y_s$.
- N_s parallel analog processing of the dot products $\boldsymbol{x}^T\boldsymbol{x}_s$ using N_s structures, one for each support vector.

For a linear SVM, the N_s scalar outputs and the bias term b enter a summation block.

The model of this time-continuous implementation is shown in *Figure 1*. The signals entering the classification phase are:

- the support vectors, denoted by "*Support Vector s*", s=1,...N_s as time-continuous signals $x_s(t)$ and their labels y_s (+1 or -1);
- the Lagrange multipliers α_s, denoted by "*alpha s*", s=1,2,...,N_s;
- the test vector to be classified $\boldsymbol{x}$, denoted by "*Xtes*t", as a time-continuous signal $x(t)$;
- the bias term b, denoted as "*Bias*".

The value of the decision function $f(Xtest)$ is read at the moment N/f_S. Label is given by the Zero threshold comparator: +1 if $f(Xtest)>0$ or -1 if $f(Xtest)<0$.

The last issue to complete the proposed formulation of the SVM classification phase is the generation of the signals $x(t)$ and $x_s(t)$ from their sampled versions $\boldsymbol{x}$ and $\boldsymbol{x}_s$, s=1,2,...,N_s. In our implementation we adopt the technique used in [3]. An illustration of the resulting signals $x(t)$ and $x_s(t)$ are illustrated in *Figure. 5*, for a particular example with N_s=2 and N=4 (the plots denoted *XTEST* and *SV*1). They can be considered periodical square wave signals, with the period N/f_S. In the ideal case of no time delay between $x(t)$ and $x_s(t)$, at the output of the integrator**,** at the moment N/f_S we will obtain [3]:

$$\int_0^{N/f_S} x(t)x_s(t)dt = \sum_{k=0}^{N-1} \int_{k/f_S}^{(k+1)/f_S} x(t)x_s(t)dt = \frac{1}{f_S}\sum_{i=1}^{N} x_i x_{si} \quad (3)$$

s=1, 2, ... N_S, that is proportional to the dot product by a factor of $1/f_S$.

III. PARTIALLY SEQUENTIAL-PARTIALLY PARALLEL ANALOG SVM CLASSIFIER

The designed analog SVM classifier is implemented in Orcad using the *MLT04* analog multiplier to carry out the multiplication, and some operational amplifier for integration, summation and comparison. In order to gain insight in the full behavior of the classifier, we choose a simple classifier with two support vectors.

The analog multiplier block that implements the feature-by-feature multiplication of each support vector with the test vector is presented in *Figure 2*. The inputs of the analog multiplier circuit, *MLT04/AD*, are denoted with *X* and *Y*, while the output is denoted with W. According with its data sheet [4], the analog input range is ± 2.5V and it operates from ± 5V supply. Output offset voltage in the *MLT04* is factory-trimmed to ±50 mV, and the scale factor is internally adjusted to ±2.5% of full scale. Input offset voltage errors can be eliminated by using the optional trim circuits (*U11A*,

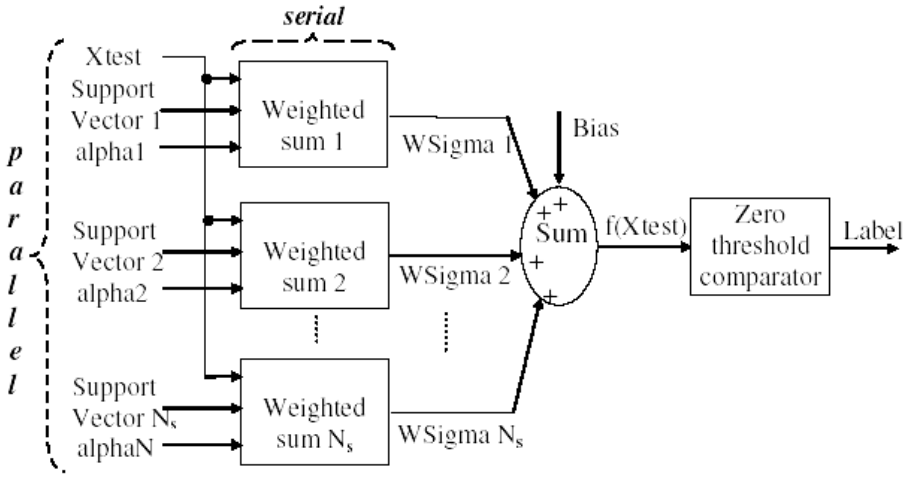

Figure 1. Block diagram of the analog SVM classifier

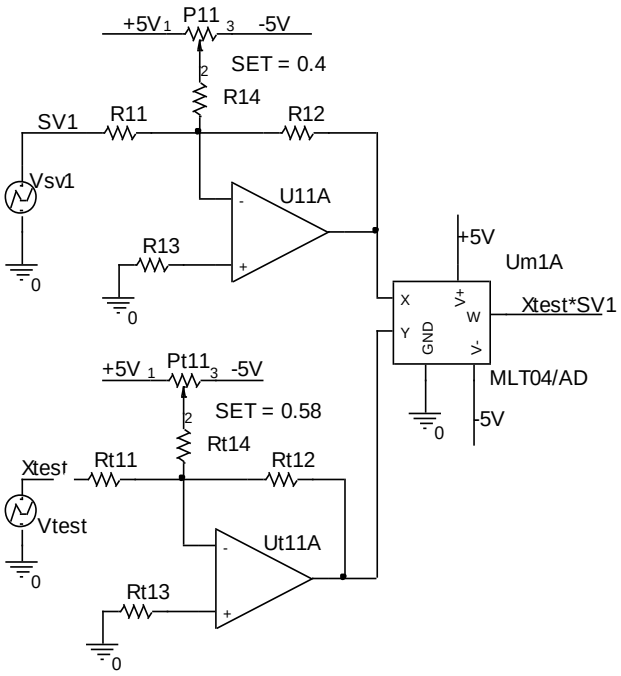

Figure. 2 The feature-by-feature multiplication

P11, R14, R11, R12, R13 for input X) in *Figure 2*.

This scheme reduces the net error to output offset, scale-factor (gain) error, and an irreducible nonlinearity component in the multiplying core. The scale factor of the multiplier being 2.5, it follows:

$$W = \frac{XY}{2.5} \tag{4}$$

We should mention here that the input voltage sources Vsv1 for support vector, and Vtest for the test vector provide constant positive values lasting 1ms for each vector feature. Both signals appear with negative values at the input of the analog multiplier due to the inverting trim circuit. Multiplying two negative values, the multiplier generates only positive signal to its output. Anyway, because of the nonlinearity of the multiplication we have trimmed the circuit to minimize the errors for an input range of [0.3V; 2.5V]. It is not a difficult task to prove the relation between the input and output signals:

$$(Xtest*SV1)(t) = \frac{SV1(t)Xtest(t)}{2.5V} \tag{5}$$

Figure 3. presents the integrator and the multiplication circuit that computes the signal WSigma1. One problem with the integrator (*U12A, C1i, R16* and *R17*) is that the output tends to wander off even for zero voltage at the analog multiplier inputs, because a very small voltage (few mV) is present at the integrator input due to the non-perfect offset trimming of the analog multiplier. This problem can be solved [4] if the integrator is zeroed periodically (in our case the period is N/f_S) by closing a switch placed across the capacitor (*M*1 field effect transistor). The integrator is an inverting one, therefore the signal at its output is denoted *–Sigma1*. The expression for *–Sigma1* is:

$$-Sigma1(t) = -\frac{1}{R_{16}C_{1i}} \int_0^{N/f_s} \frac{(Xtest*SV1)(t)}{2.5} dt \tag{6}$$

Since the signal $(Xtest*SV1)(t)$ is piecewise constant, the result at the end of integration is:

$$-Sigma1(N/f_S) = -\frac{1}{R_{16}C_{1i}} \frac{1}{f_S} \frac{Xtest*SV1}{2.5} \tag{7}$$

To make *-Sigma1* to be exactly the dot product, we set the value of 2.5ms for the time constant of the integrator ($R_{16}C_{1i}$) and the value of 1ms for $1/f_S$. Thus at the end of the integration, we get:

$$-Sigma1(N/f_S) = Xtest*SV1 \tag{8}$$

The right part of the circuit in *Figure 3*. (*U13A* and the components around) performs the multiplication of the

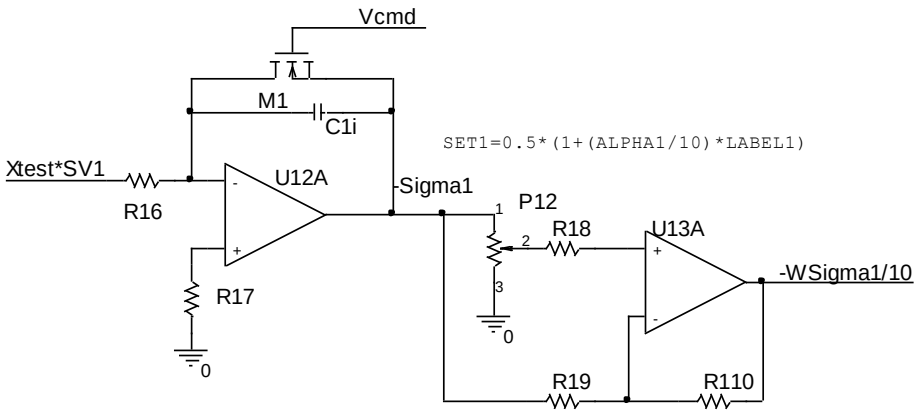

Figure 3. The integration and the multiplication circuits.

signal -*Sigma1* with the label and the Lagrange multiplier of the support vector. As one can find, the gain of this amplifier can be set in the range [-1,1] using the *P12* potentiometer (for *R19=R110*). The sign of the gain sets the Label (-1 or +1), while the gain magnitude sets the value of the Lagrange multiplier, Alpha. The current implementation assumes Lagrange multipliers Alpha less than 1.

The output signal is:

$$-WSigma1 = -Sigma1(2SET - 1) \quad (9)$$

where *SET* is the fraction of the potentiometer between its tap and its end connected to the ground. For a specific *Alpha*, the *SET* is computed as

$$SET = 0.5(1 + Alpha1 \cdot Label1) \quad (10)$$

If the values of Alphas after the training phase are larger, we should scale them in the range [0;1]. Using the same scaling factor for all *Alphas* and for *BIAS*, the result is a scaled version of the decision function of the SVM, keeping the sign of the initial non-scaled version.

The next circuits are the summing and the comparator, as one can see in *Figure 4*. The first one adds the values of -*WSigma1*, -*WSigma2* and –*BIAS*. The *BIAS* value is set by the the *Pbias* potentiometer. If *R2=R3=R4=R5*, the decision function computed to the output of the inverting summing circuit is:

$$f(t) = -(-WSigma1(t) - WSigma2(t) - BIAS)$$
$$f(t) = WSigma1(t) + WSigma2(t) + BIAS \quad (11)$$

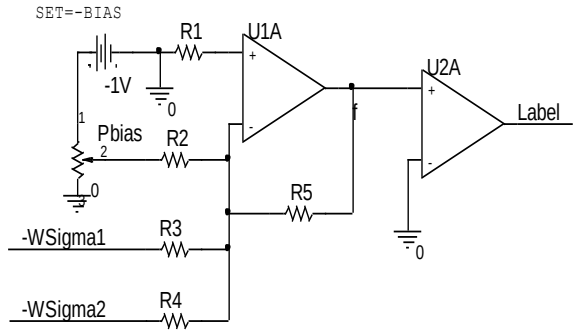

Figure 4. The summing and the comparator circuits

Finally, the simple non-inverting comparator (*U2A*) gives the class label. It responds with a positive voltage (close to the positive supply) for a positive example or with a negative voltage (close to the negative supply) for a negative example. Our proposed implementation can be easily extended for more support vectors by replicating the computing channels and adding inputs to the summing block. Because the involved *MLT04* circuit has four channels we can build a four-support vector SVM classifier with only one such analog multiplier integrated circuit.

IV. EXPERIMENTAL RESULTS

We tested our implementation on a standard data set for classification tasks, namely, the IRIS data. Each pattern is described by four features: the petal length and width; the sepal length and width. There are 3 classes of irises: Setosa, Versicolor and Virginica. We have considered here the binary classification in the class Setosa vs. the other 2 classes. For the SVM training we used Steve Gunn's Matlab application [6]. The IRIS data set contains 75 training vectors.

The range of the iris flower features is [0.1;7.9]. Because in our implementation the range of input voltages of the analog multiplier is [0.3V; 2.5V], we use a data conversion in the form:

$$converted\ data = 0.27179 + 0.28205 data \quad (12)$$

We trained on this converted data a linear SVM with the error penalty parameter C=10. After training we got 2 support vectors. We classify the standard IRIS test set using two classifier implementations: the software Steve Gunn's implementation (considered here as reference classifier, and denoted SSG) and our **h**ardware **a**nalog **t**ime **c**ontinuous SVM implementation (denoted as HATC). The values of the decision function in the two implementations and the "target" classification results are given in *Table 1*, for ten test vectors. In our implementation we scaled the values of *Alphas* and *BIAS* by a factor of 0.1, to bring them in the range [0;1]. As a result the decision function is also scaled by a factor of 0.1. In order to have a direct comparison between the numerical results we present the values of the decision function resulted in the SSG implementation divided ten times, *f(Xtest)*/10. The classification labels, given by the sign of *f(Xtest)*, always match the target ones, denoted there as *Label*.

Table 1. Classification results

Xtest	***f(Xtest)*/10 SSG**	***f(Xtest)*/10 HATC [V]**	***Label***
1.	0.1107	0.1018	1
2.	-0.3407	-0.3496	-1
3.	-0.4557	-0.4645	-1
4.	-0.1179	-0.1268	-1
5.	0.0842	0.0751	1
6.	0.0928	0.0838	1
7.	0.1728	0.1617	1
8.	0.0770	0.0681	1
9.	-0.1269	-0.1358	-1
10.	-0.2877	-0.2966	-1

As one can notice by examining *Table 1*, the differences in the real value of the decision function computed by SSG implementation and our HATC implementation are very small. In fact we can observe that for all the test vectors the values of the decision function are approximately with 9mV less than the value corresponding to the SSG implementation. For example for the 2^{nd} test vector we

obtained -0.3496V instead of -0.3407, meaning a difference of 8.9mV; for 6th test vector we obtained 0.0838V instead of 0.0928, meaning a difference of 9mV; for 9th test vector we obtained -0.1358V instead of -0.1269, meaning a difference of 8.9mV. This seems to be systematic departure from the right value, so it can be considered as a negative offset voltage for the decision function. If a greater accuracy is necessary we have to compensate this offset voltage. As a whole, we can conclude that our HATC implementation is very accurate, all the test vector being correctly classified.

The computational details in each step of the algorithm can be seen in *Figure 5*. This figure illustrates the waveforms for all the inputs, intermediate and output signals during the classification of the 4th test vector in *Table 1*. The signal for the test vector *V(XTEST)* is quite similar with the signal for the second support vector *V(SV*2*)* but different from the signal of the first support vector *V(SV*1*)*. We can anticipate a membership of the test vector at the same class as *SV*2.

After the feature-by-feature multiplication of the test vector with each support vector we obtain the signals *V(XTEST*SV*1*)*, respectively *V(XTEST*SV*1*)*. We mention here that these signals contains the scale factor of the analog multiplier. As we mentioned earlier, the time constant of the integrator compensates the scale factor of the analog multiplier. At the time moment N/f_S=4*ms* (*N*=4, the number of each vector features; f_S=1kHz, the sampling frequency) we have the scalar values of the kernel function: *V(–SIGMA*1)=-4.955V, respectively *V(–SIGMA*2*)*=-5.377V. The minus sign appears due to the inverting integrator. The signal *V(-WSIGMA*1/10*)*, keeps the same sign with the *V(-SIGMA*1*)* because the label of the first support vector is *LABEL1*=+1, but it is weighted with one tenth of the corresponding Lagrange multiplier (0.1·8.69220391247454). The signal *V(-WSIGMA*2/10*)*, changes the sign compared with *V(-SIGMA*2*)* because the label of the second support vector *is LABEL*2=-1, and it is weighted with one tenth of the corresponding Lagrange multiplier (0.1·8.69220394804136). The time variation of the decision function *V(F/*10*)* gives important information about intermediate memberships of the test vector after taking into consideration only the first features of the test vector, after each time period. In accordance with the first two features (time moment *t*=2ms), *V(F/*10*)*>0*V*, *V(LABEL)*=+14.3V (corresponding to *Label*=+1), so the test vector appears to belong to the Setosa class. However after considering the third feature (*t*=3s), *V(F/*10)<0, *V(LABEL)*=-14.3V so the test vector does not belong to Setosa. The final result (at *t*=4ms) confirms this classification: *V(F/*10*)=-0.1268V*, *V(LABEL)*=-14.3V (corresponding to *Label*=-1).

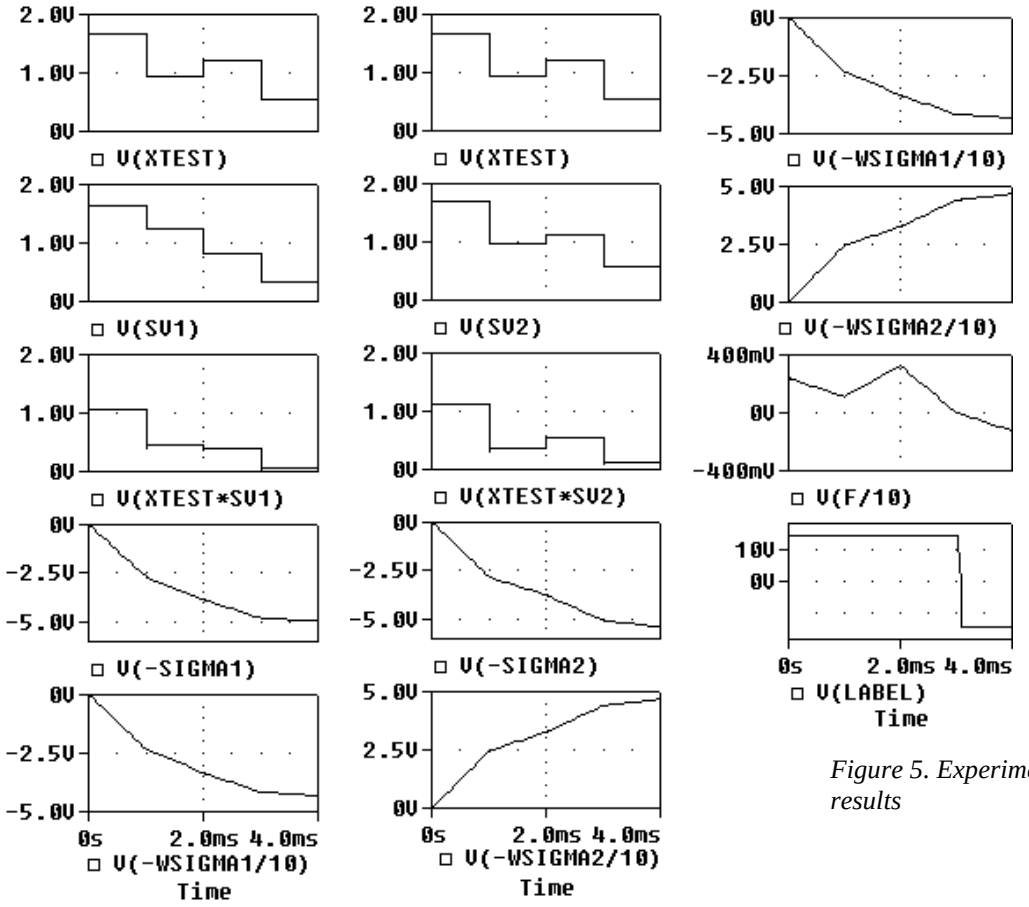

Figure 5. Experimental results

CONCLUSIONS

In this paper we proposed a partially sequential-partially parallel implementation of the SVM classification phase based on a analog multiplier. This new implementation represents a good compromise solution between the duration of the classification phase and its complexity in analog hardware implementation, especially in large data spaces. The proposed algorithm is based on the description of the feature vector **x** and support vectors $\mathbf{x}_s$ of the SVM as analog signals x(t) and $x_s(t)$, making possible to implement the kernel in a simple analog signal processing fashion. The equivalence of the standard formulation and the proposed implementation was proven using an SVM classifier with two support vector implemented with a MLT04 analog multiplier and operational amplifiers. The accuracy of the classifier was then tested through experiments on the IRIS dataset. The decision function was correctly computed for all the test vectors, with a systematic small offset voltage around 9mV; all the test vectors were correctly classified. The graphical illustration of the time-continuous classification is useful to understand the SVM classification process, and also can help a lot in selecting only the important features of the vector to reduce the computing time. It is worth to mention that the complexity of the hardware increases with the number of support vectors of the classifier, as in the existing hardware implementations, but does not increase at all with the data dimension. This is a great advantage compared with the existing implementation, where the complexity increases in direct ratio with the data dimension.

REFERENCES

[1] V. N. Vapnik, *Statistical Learning Theory*, J. Wiley, N.Y., 1998.

[2] R. Genov, S. Chakrabartty, G. Cauwenberghs, "Silicon Support Vector Machine with On-Line Learning", *Int. J. of Pat. Recog. and Artificial Intelligence*, 2003, pp. 385-404.

[3] G. Oltean, Mihaela Gordan, "Towards Analog Implementation of Support Vector Machine: A Time-continuous Formulation of the Classification Phase", *Proceedings of the 12th International Conference Mixed Design of Integrated Circuits and Systems MIXDES*, 2005, pp.549-554.

[4] *** MLT04, Four-Channel, Four-Quadrant Analog Multiplier Data Sheet, Analog Devices.

[5] Horowitz, P., Hill, W. – *The Art of Electronics*, Cambridge University Press, UK, 1997.

[6] S.R. Gunn, MATLAB Support Vector Machine Toolbox, March1998, http://www.isis.ecs.soton.ac.uk/resources/svminfo.