FPGA Implementation of SVM for Nonlinear Systems Regression

Intissar SAYEHI
University of Tunis Elmanar, Faculty of Mathematical, Physical and Natural Sciences of Tunis
Laboratory of Electronics and Microelectronics, (E. µ. E. L),
FSM, Monastir, Tunisia

Mohsen MACHHOUT
University of Monastir, Faculty of Sciences of Monastir
Laboratory of Electronics and Microelectronics (E. µ. E. L)

Rached TOURKI
University of Monastir, Faculty of Sciences of Monastir
Laboratory of Electronics and Microelectronics (E. µ. E. L)

***Abstract*—This work resumes the previous implementations of Support Vector Machine for Classification and Regression and explicates the different methods and approaches adopted. Ever since the rarity of works in the field of nonlinear systems regression, an implementation of testing phase of SVM was proposed exploiting the parallelism and reconfigurability of Field-Programmable Gate Arrays (FPGA) platform. The nonlinear system chosen for application was a real challenging model: a fluid level control system existing in our laboratory. The implemented design with fixed point precision demonstrates good enough results comparing with the software performances based on the Normalized Mean Squared Error. Whereas, in term of computation time, a speed-up factor of 60 orders of time comparing to MATLAB results was achieved. Due to the flexibility of Xilinx System Generator, the design is capable to be reused for any other system with different data sets sizes and various kernel functions.**

Keywords*—Machine learning; nonlinear system; SVM regression; Reproducing Kernel Hilbert Space (RKHS); MATLAB; Field-Programmable Gate Arrays (FPGA); Xilinx System Generator

I. INTRODUCTION

The support vector machine is a machine learning created by Vapnik at the 60's. It was created first for classification tasks then extended to regression. The most important advantage in this method that is applicable to different fields are like medicine biology, signal processing, sensor networks, computer sciences, etc.

The difficult challenge in the use of the SVM method is to compromise between the model performances and the data sets size. There from the need to hardware platforms that accelerate the computation time and provide a flexible support for classifying or regressing new systems.

This paper treated the previous hardware implementations of support vector machine on Field-Programmable Gate Arrays (FPGA) for classification and regression and explains the different approaches adopted for developing the SVM architecture. The FPGAs devices offer many advantages like concrete development tools, simple reprogram ability and quick development time. Furthermore, parallelism can be attained, that is a benefit above other devices, like microcontrollers and DSPs.

The majority of implementations of SVM were targeted to classification task for simple and specific problems. However the SVM regression task still abandoned and neglected.

In this present, we propose a hardware design on FPGA for nonlinear systems regression.

The arrangement of the article is as pursues. In the second section the theoretical basis of nonlinear systems identification in the Reproducing Kernel Hilbert Space (RKHS) space was described. In Section 3, the Methods of SVM implementations on FPGA were presented with the related works. In Section 4, the FPGA designing tools were explained. In Section 5 our SVM Implementation approach and the Parameters Selection were presented with results and plots for the regression of fluid level control system. Finally we compare our work to similar ones and conclude with Conclusion.

II. NON LINEAR SYSTEMS IDENTIFICATION IN REPRODUCING KERNEL HILBERT SPACE

The identification of linear systems is accomplished via mathematical representations on the bases of vibration measurements. Thus, the relation between the input and output of a system, called transfer function, stays stable at all excitation levels. Accordingly, the mathematic model acquired at one operating point can be generalized for predicting the system behavior at another operating point. Whereas, it is not the same case for nonlinear systems because it is hard and complicated to find a general mathematical model describing the system by relying on the system identification only at a particular excitation level.

The difference between linear and non-linear systems can be explicated by Fig. 1, that for non-linear systems the transfer function is not independent of the input.

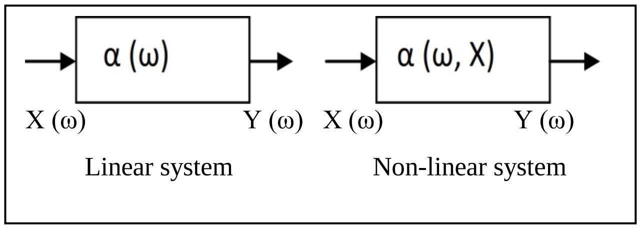


Fig. 1. Distinction between linear and nonlinear system.

It exist many types of nonlinear models like state-space models, Input/output-models, block-oriented models, etc. it didn't exist a nonlinear universal model suitable for all appliances but it depends on each one. Consequently, diverse approaches for identifying and modeling nonlinear systems were conceived. Essentially two categories can be distinguished: parametric models and nonparametric models. The next figure clarifies the different constitutions of these categories.

Another part of researchers were also interested by the system identification field by creating diverse techniques in machine learning allowing nonlinear systems identification like k-Nearest Neighbors and Regularization networks [1]-[3] .

In next paragraphs, we introduce the mathematical foundation of learning machine and explain the functionality of support vector machine method.

A. Statistical Learning Theory (SLT)

The goal of the Statistical Learning Theory [4] is to obtain a function f modeling a given system since a set of observed data $\mathrm{O}=\{(\mathrm{x_i},\mathrm{y_i})\}_{\mathrm{i}=1}^{\mathrm{N}}$ composed of inputs x_i and outputs y_i . This function has to repeat the process behavior by diminishing the functional risk presented by this expression:

$$R(f)=\int_{X,Y} V(y, f(x))P(x,y)dxdy \tag{1}$$

The expression V(y,f(x)) is named cost function. It computes the deviation between system output yi and the estimated output f(x). The couple (X,Y) is composed of random vectors and of the independent samples (x_i, y_i) . The risk R(f) can't be approximated caused by unknowing P(x,y). To resolve this problem we have to ease the following expression:

$$R_{emp}(f)=\frac{1}{N}\sum_{i=1}^{N} V(y_i, f(x_i)) \tag{2}$$

However, minimizing frankly $\mathrm{R_{emp}}$(f) in the functions space H don't give the best approximation of R(f) minimization and could guide to over fitting. As a resolution, Vapnik presented the theory of structural risk minimization (SRM). It penalizes empirical risk via a function that approximates the complexity of the retained model.

This guides to minimizing the constraint defined by the following expression:

$$\min_{f\in H} D(f)=\min\left(\frac{1}{N}\sum_{i=1}^{N} V(y^{(i)}, f(x^{(i)}))+\lambda\|f\|_H^2\right) \tag{3}$$

The first word measures how the function f fits a given data and the second word is the squared norm of f in the RKHS space H that controls the complexity (smoothness) of the solution. The parameter λ is the regularization parameter that equilibrium the tradeoff among the two terms.

The regularity of the solution is most important and not the value of λ. Whereas it is not evident to minimize the constraint (3) on whichever arbitrary function space H, whatever is it with finite or infinite dimension. Therefore, to overcome this difficulty, we will consider the space H as a RKHS.

B. Reproducing Kernel Hilbert Space (RKHS)

We suppose that X the random variable is evaluated in the space $E \subset \mathbb{R}^d$ and we suppose that exists a function K called kernel function: $K: E^2 \to \mathbb{R}$. It is symmetric and positive definite. In this case, there is a function $\phi: E \to H$ that:

$$K(x,x'))=\langle \phi(x), \phi(x')\rangle_H \tag{4}$$

H is the Reproducing Kernel Hilbert Space (RKHS) [5] of kernel *K*. This space has some rigorous properties:

- ✓ $\forall x \in E$ et $f \in H$ $\langle K(x,.), f\rangle_H = f(x)$ (5)
- ✓ Due to represented theorem the resolution of the optimization problem presented by (3) in this space is given by :

$$f_{opt}=\sum_{i=1}^{N} a_i K(x_i,.) \tag{6}$$

TABLE I. KERNEL FUNCTIONS

Kernel function	**Mathematical expression**	**Parameters**
Linear kernel	$K(x,x'))=x\times x'$	-
Polynomial kernel	$K(x,x'))=(1+\langle x,x'\rangle)^{\eta}$	$\eta\in\mathbb{N}^*$ and $\langle x, x'\rangle$ is an Euclidian scalar product.
Sigmoid kernel	$k(x,y)=\tanh(\alpha.x^T.y+c)$	α and c are adjustable
Radial Basis Function (RBF) kernel	$K(x,x'))=e^{\frac{\|x-x'\|^2}{2\sigma^2}}$	σ is a real positive parameter
Laplacian kernel (ERBF, Extended RBF)	$K(x,x'))=e^{-\gamma\|x-x'\|}$	γ is a real positive parameter

It exist a variety of kernel functions that could be considered in Table 1.

C. Support Vector Machine for regression

SVM is a supervised learning model founded on the Vapnik and Chervonenkis learning theory. It was first developed for classification problems then extended to regression tasks.

For SVM regression, the goal is to find a model for the data set D={(x1,y1),(x2,y2),…,(xn,yn)} that matches the input xi to the real output yi (with $x_i \in \mathbb{R}^l$ and $y_i \in \mathbb{R}$).

By resolving the following quadratic programming problem with linear restrictions and an ε-insensitive loss function:

$$\min_{\alpha,\alpha^* \in \mathbb{R}^l} \text{imise} \frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)K(x_i, x_j) + \varepsilon\sum_{i=1}^{n}(\alpha_i^* + \alpha_i) - \sum_{i=1}^{n}(\alpha_i^* - \alpha_i)y_i \quad (7)$$

$$\text{St}: \ 0 \le \alpha_i^*, \alpha_i \le C \ \text{ for i=1,…,n } \ \sum_{i=1}^{n}(\alpha_i^* - \alpha_i) = 0$$

where K(x$_i$,x$_j$) is a kernel function, C is the regularization term and ε is a positive constant presenting an insensitive region in the interior of which the training errors are unseen. C and ε are predefined constants. The feed-forward evaluation function of a new, unlearned vector x is:

$$y(x) = \sum_{i=1}^{Nsv}(\alpha_i^* \alpha_i)K(x_i, x) + b \quad (8)$$

The parameters α_i, α^*_i and b are calculated in learning phase. As in the classification model, just the resulting support vectors are used in the feed-forward phase.

III. Methods of SVM Implementation on FPGA

The related hardware implementations of SVM model on FPGA was accurately reviewed. In this paper we are focused in certain class of SVM implementations that implemented just the testing phase on FPGA. Unfortunately those for SVM regression were rare and exceptional. Most of designers needed to implement classifier for different applications whereas we found only three works [6]-[8] that implement a design for both classification and regression. The SVM for regression has the same importance as SVM for classification but is infrequently employed due to the complexity of the feed forward function. Next paragraphs give a consistent recap of different techniques and architectures employed for SVM implementation on FPGA.

A. Parallel Systolic Array Architecture

In different fields of sciences, the operations involving important linear system of equations like matrix algebra are indispensable. Consequently, the need of fast and speedy computers equipped with efficient software programs is crucial and increasing. Whereas, the main disadvantage of a general-purpose computer is the limited memory space for big matrices computing. To avoid this problem, novel methods and approaches have to be invented to benefit simultaneously of highly parallel computational machines.

The solution was the association of a big number of same processing elements (PEs) that rhythmically compute and pass data to neighboring connected PEs. The produced set of well ordered PEs connections corresponds to the Systolic architecture that can be arranged in a linear or two-dimensional array with rectangular or hexagonal geometry.

The systolic array could be employed as a coprocessor combined with a host computer that pass data through the PEs and the final result is came again to the host computer. As in Fig. 2, this operation is similar to the flow of blood throughout the heart called "systolic".

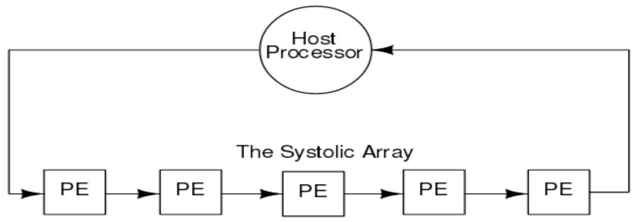


Fig. 2. Systolic system machanism.

This arrangement is very suitable for VLSI technology that offers an exceptionally high operating with low cost array of speedy computational processors. It was broadly implemented on FPGA to attain high levels of parallelism, that was exploited by various SVM implementations.

R. Patil et al. [9] employed this architecture for implementing a SVM multiclass classifier. The hardware was Xilinx Virtex-6 FPGA for the recognition of facial expression. The training phase computed by MATLAB. Thanks to a power optimization of the FPGA design based on the difference-based partial reconfiguration technique, the power decrease up to 3 to 5% was attained by using Xilinx EDA tool.

C. Kyrkou et al. developed an SVM classifier for object detection based on systolic array architecture [10]. The design can be expanded and adjusted to convene multiclass classification and various applications. Many tasks of object detection like face, walker, and car were done. Simulation consequences proved a high performance of 40, 46, 122 fps for three applications, with no precision loss in comparison with the precision of software detection of the SVM model executed in MATLAB (77, 76, 78%).

B. Multiplier-less Approach

The Multiplier-less techniques were needed to diminish the implementation cost because the multipliers are the mainly costly blocks in term of surface occupation. Therefore, many researchers have hardly worked to make the multiplication simpler and faster by applying the fact that multiplication by a power-of-two could be achieved by simple shift and add operations. The number of these operations depends on the design restrictions. There are a number of conventional representations to speed up multiplication. One is by reducing the number of operands to be added; the other is by adding the operands faster (accelerating accumulation) [11]. Most designers combine and profit from the two methods to reduce significantly the number of operands employed in the hardware. In the work [12], the authors benefit from this technique and aiming a diminution in hardware complexity and power consumption by implementing simplified multiplier-less kernel using shifts and add operations as an alternative to traditional vector product kernel for classification. The implementation of SVM classification was performed on the modern Xilinx Virtex-7 FPGA.

They presented a different approach of applying the CSD and CSE representation methods for vectors data to decrease the number of needed adders to reduce the hardware complexity. Three classifiers were implemented to compare it against other three implemented classifiers using the conventional vector product kernel. The power reductions of 1, 2.7, and 3.5% were achieved by the proposed CSD-based multiplier-less kernel against the vector product kernel relating to resources utilization.

C. Dynamic Partially Reconfiguration

There are two methods to modify the Hardware functionality on FPGA. First, the Static reconfiguration which consists to shutting down the application then downloading the new configuration and restarting the implementation. Second, the Dynamic Reconfiguration permits varying hardware functionality on FPGA without taking the purpose offline. This offers a flexibility to adjust the hardware online and to gain a lot of time. The modification can be either total or partial according to the need of designer. In total reconfiguration the configuration bitstream, affords the information concerning the chip and it arranges whole FPGA. In partial reconfiguration, just a part of the platform is reconstituted, whereas the rest maintain operating securely with respect to the development procedure. The Dynamic Partially Reconfigured approach (DPR) grants the modification on a selected section of the FPGA while the other sections stay working without necessitating to turning off. This great improvement is excellent for real time embedded systems when the shutting down of the system is expensive and detriment during system runtime. Moreover, DPR diminished significantly power consumption and decreasing reconfiguration time.

This practice was utilized by H. Hussain et al. [13] for implementing SVM classifier for bioinformatics applications. It was implemented on an old FPGA panel; Xilinx ML403, where the kernel calculation was implemented using two pipelined stages. An acceleration up to 85x was accomplished above a corresponding GPP software execution using MATLAB bioinformatics toolbox DPR was applied to change the diverse parameters of SVM, it was 8x more rapidly than reconfiguring the entire of FPGA.

D. Common Pipelining Technique

The pipelining technique is a technique implementing a form of parallelism with a single processor. It accelerates the central processing unit throughput at a certain clock rate by performing multiple operations at the same time. The fundamental training cycle is broken in a series named a pipeline. Pipelining searches to let the processor works on as many instructions as there are dependent steps. It augments instruction throughput however does not diminish the required time to end one instruction.

The researchers in works [14], [15] wanted to compare the performances of FPGA and GPU implementations of a human skin SVM classifier against the software performances. The critical hardware composes of FPGA were designed using HDL in a completely pipelined organization, even as the other elements like FIFO and interfaces were implemented in HLL. The implementation results confirmed the excellence of the implemented fully pipelined FPGA architecture on GPU and CPU for a small number of image pixels, while the GPU implementation was the fastest for a big number of pixels. The advantage of FPGA implementation is that consumes less power than the GPU implementation [15].

Y. Ago et al. [16] employed a new fully pipelined DSP architecture on FPGA for accelerating SVM classification. The proposed design was executed on Xilinx Virtex-6 FPGA with different types of kernel functions; sigmoid, polynomial, and RBF kernels. Consequently, an important throughput of 2.89x106 times per second for classifying 128-dimension feature space running at 370.096 MHZ was obtained. Other implementations intended to develop the common pipelined fashion for accomplishing powerful designs.

Whereas, a combined circuit was designed in a parallel architecture with two-stage pipeline for linear and non-linear SVM classification [17] in order to diminish the circuit size by sharing multipliers and adders necessary for inner product computation. The proposed circuit was synthesizing with 65nm standard cell library, representing 661,261 gates with 152 MHz maximum operating frequency. Moreover, high performance was attained from handing out up to 33.8 640x480 image frames per second.

E. CORDIC Algorithm

The CORDIC algorithm is a fundamental iterative algorithm using a fixed vector rotation technique to calculate sequentially the trigonometric functions. The entire conception orbits around employing just a simple shifter and adder to simplify the implementation of the CORDIC algorithm for computing complex functions. The CORDIC algorithm was originally presented by J.E. Volder [18] for implementing fundamental mathematical functions like the multiplication, division and trigonometric functions. It was helpful for diverse domains like neural networks, video and image processing, etc. For majority of applications the CORDIC algorithm offers a speed-up of time and reduction of power consumption. SVM method for both classification and regression were implemented by M. Ruiz-Llata et al. [6] on FPGA using the CORDIC iterative algorithm. The implemented system consumed 3/4 of the FPGA logic (Cyclone II) and an exterior memory was used for storing support vectors leading to 2ms limitation in the classification speed, with an error rate of 4%.

Another FPGA implementation of fast SVM presented by J. Sarciada et al. [19] based on CORDIC algorithm for kernel calculations. The implemented system achieved speed improvement over their previous CORDIC circuit implemented in [20] with a factor of 6, with limited hardware resources utilization.

IV. FPGA Designing Tools

Field-Programmable Gate Arrays (FPGAs) is composed of configurable logic blocks (CLBs) that can be reprogrammed to realize different functions in few seconds. The flexibility offered by the FPGA goes with the increasing programming complexity. Consequently there is a critical necessity for high level fast prototyping systems that can help designers and eases the mapping from algorithm to hardware. The algorithms are classically written and tested via MATLAB code or Simulink model based environment, and there are a number of tools that convert such algorithms to a hardware description language such as AccelDSP Synthesis Tool from Xilinx, Simulink HDL Coder from Mathworks, C-based High Level Design Tools and System Generator for DSP. Briefly, these tools are explained at the next paragraph.

AccelDSP Synthesis Tool is a high level tool particularly designed for Digital Signal Processing (DSP) applications. The principle of AccelDSP is to translate a MATLAB floating-point design into a hardware implementation that objects FPGAs. AccelDSP automatically creates bit-true and cycle-accurate HDL codes which are complete to synthesize, implement and map onto FPGA hardware. AccelDSP generated designs result in inefficient architectures in terms of area and timing compared to hand-coded results.

The Simulink HDL Coder is a high level design tool which generates HDL code from Simulink models and State flow finite state machines. It can provide also interfaces to combine manually-written HDL codes, HDL Co-simulation blocks and RAM blocks in its environment. Whereas, not the whole Simulink included blocks are supported. Embedded MATLAB Function Block has its own limitations and do not support all the operations such as nested functions and use of multiple values in the left side of an expression.

The design C-based high level design tools [21] are used for automatic hardware generation offering a quicker path to hardware with a low cost comparing to traditional methods. It expresses parallelism through variations in C (pseudo-C) or compiler or both. Ideally, it is best to use pure ANSI-C without any variation in C and exploit parallelism through compiler that ports C code into hardware; therefore a user does not need to learn a new language.

System Generator is a high level design tool designed by Xilinx to be used in model-based design environment and implemented in FPGAs. Simulink provides a powerful component based computing model including several different blocks to be connected together by the user for designing and simulating functional systems. System Generator provides similar blocks which are used and connected the same way Simulink blocks does but target FPGA architectures to design discrete time systems which can be synchronous to a single or more clocks. The simulation results of the designed systems are bit and cycle accurate which means simulation and hardware results are exactly match together. System Generator is the best tool provided for MATLAB code environment because it's a "push button" transition from specification to implementation.

V. Implementation of SVM Regression

A. *Approach and the Parameters Selection*

A variety of practices for hardware implementations have been developed for improving the online SVM testing phase on different FPGA devices. The idea is to train SVM model first offline on software (MATLAB), and then the trained data are extracted to be used for online regression on hardware. Only the resulting support vectors are used in the feed-forward phase. The feed-forward estimation function of a new, non-learned vector x is:

$$y(x) = \sum_{i=1}^{Nsv} (\alpha_i^* \alpha_i) K(x_i, x) + b \tag{8}$$

Where parameters αi, α_i^* and b are given in the learning phase. $K(x_i,x_j)$ is the kernel function that can be polynomial functions or Gaussian functions. Our system uses the polynomial function because this kernel significantly simplifies the SVM feed-forward phase computation in constrained hardware while conserves good classification performance with respect to the system nonlinearities.

The parameters to be fixed prior the training step are the parameter of the kernel which is the degree of polynomial kernel, the regularization parameter C and the ε parameter of the ε-insensitive loss function. All these parameters are chosen to be used in the fixed-point arithmetic. To locate C, and the ε parameter in regression, we use an iterative training strategy with the goal of minimizing errors while keeping a reduced number of support vectors.

The basic hardware architecture to perform (8) is represented in Fig. 3.

The inputs parameters are: the testing vector and the support vectors. The calculation of the prediction function is realized through a kernel function processing block that acquires the input parameters and then calculates the Gramian matrix to be multiplied by the Lagrange Multipliers.

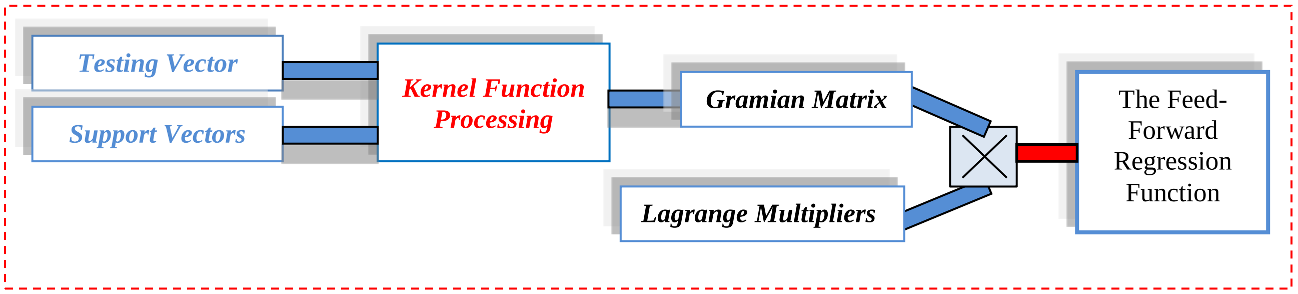


Fig. 3. Hardware architecture of SVM testing phase.

The Gramian matrix $G \in \mathbb{R}^{N\times N}$ is like that:

$$G_{ij} = (K(x_i, x_j)), i, j = 1,...,N \tag{9}$$

N is the number of observations and K is the kernel function. It can be selected as sigmoid or polynomial kernel. The computation of Gramian matrix is done in the training and testing phase. As the input vector X can be 1-Dimensional or 2-Dimensional Array, we suggested a streaming Approach to calculate the Gramian matrix.

This approach exploits the FPGA parallelism to automatic compilation of software programs into hardware. Effectively, three fundamental approaches are distinguished to automatic compilation of software into hardware.

First approach is to find an accessible parallel programming model. Then programmers map a program written in it onto hardware [22]. This approach permits to establish parallelism, however many problems like synchronization, deadlocks and starvation have to be arranged.

A different approach, the behavioral synthesis compilers those investigate programs written in a high-level sequential language, for example C, and challenge to extort instruction-level parallelism by analyzing dependencies in the middle of instructions, and mapping in reliant instructions to parallel hardware components. Various compilers have been accomplished, like C2H from Altera that is entirely incorporated inside their SOPC design flow. The major difficulty with such approach is that the overall of instruction-level parallelism in a classic software program is limited. Therefore, constantly have to reorganize their code and without a doubt control hardware resources, like mapping of data to memory units.

The third approach is to exploit a sophisticated language that permits to the programmers to express parallelism without be troubled about synchronization and associated matters. This type of languages is based on the streaming paradigm articulated on data that are collected into streams [23] similar to arrays, but with a mutually independency between the elements. In our work this approach is entirely used to execute all the testing phase on the FPGA.

In next paragraph the experimental process is described with explanation of the implementation steps.

B. Description of the Fluid Process and Results

The process subject to regression is a fluid level control system consisting of two cascaded tanks with free outlets fed by a pump. The water is transported by the pump to the upper of the two tanks. The process is depicted in Fig. 4.

The input signal to the process is the voltage applied to the pump and the two output signals consist of measurements of the water level of the tanks. Since the outlets are open, the result is a dynamics that varies nonlinearly with the level of water. The process is controlled from a PC equipped with MATLAB interfaces to the A/D and D/A converters. All data was collected in open loop experiments using zero-order hold (ZoH) sampling. The data was recorded from the cascaded tanks and collected in a data file. The sampling period of 4.0 s provides 7500 samples of input-output data for both the upper and lower tank.

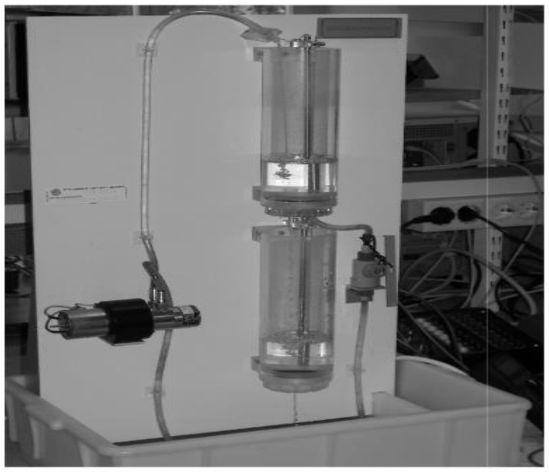


Fig. 4. Fluid process composed of two cascaded tanks.

To construct the SVM regression model for the fluid level control system, 2000 observations taken for the training phase and the validation phase was performed on 1000 new observations.

Firstly, the optimal parameters for training phase like the regularization parameter C and the ε–insensitive loss function are obtained using MATLAB with an iterative training approach to reduce computing errors while maintaining a reduced number of support vectors. After this preparation, it is possible to prove the efficiency of the modeling parameters by testing it with novel data. The type of kernel used: polynomial and sigmoid. The optimal parameters λ and σ of the machine learning are mentioned in Table 2:

TABLE II. PARAMETRERS SELECTION

Parameters / Method	**Kernel function**	**Optimal parameter**	**Trade-off parameter**	**ε-insensitive loss function**
SVM Regression	Polynomial	σ=3	C=100	$\varepsilon = 0.01$
SVM Regression	Sigmoid	α =10, c=1	C=1000	$\varepsilon = 0.01$

Secondly and according to this table, the resulting support vectors were six (for polynomial kernel). The basic hardware architecture to perform this equation:

$y(x) = \sum_{i=1}^{Nsv} (\alpha_i^* \alpha_i) K(x_i, x) + b$ is represented in Fig. 5. It is composed of input vector X, Nsv parallel support vectors SV blocks, a kernel processing block, a Gramian matrix block, Lagrange multipliers block , FIFO buffers and memory buffers.

The testing vector X is passed through the FIFO to be streamed and then calculated by the kernel functions and support vectors. To benefit from data parallelism, the number of streams is identical to the number Nsv of suport vectors because stream elements are independent. The number of streams is often limited by the accessible hardware resources.

The objective of our research is to benefit from the powerful parallelizing of FPGA and the flexible software programming. The design user has only to choose the system to be predicted and the kernel type suitable to the application, and then in offline calculate the support vectors that will be used for feed forward function on hardware.

In Fig. 5, the system generator project for implementing the testing phase is presented.

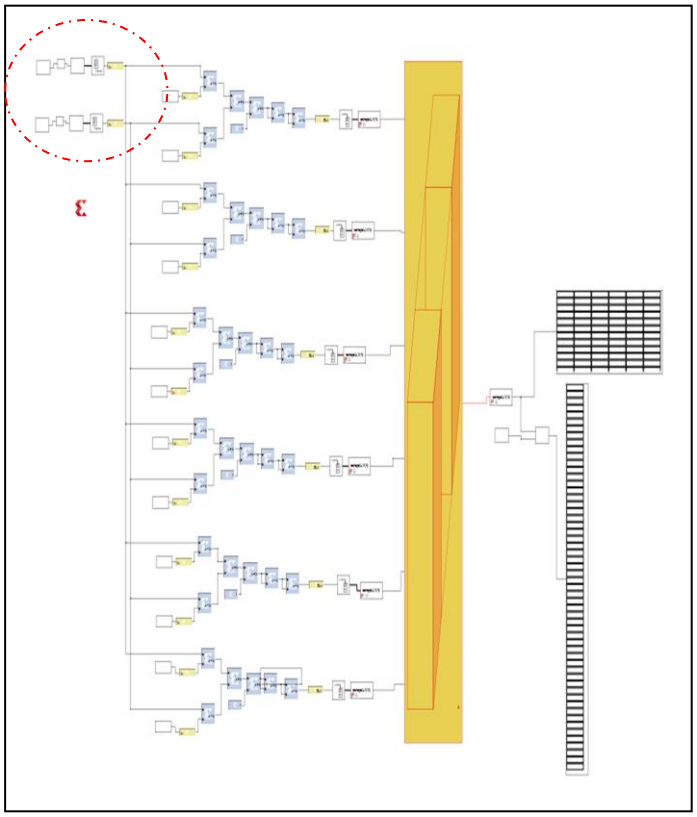


Fig. 5. System generator project for computing the testing phase of SVM.

As mentioned in Fig. 6 the regression vector is 2d-vector: $x(i)=(y(i-1),y(i-2))$ The computation of this type of vectors is generally difficult in hardware. Consequently, the streaming approach is used for this vector. The two columns of the regression vector are streamed in parallel and simultaneously computed by the kernel processing function and support vectors. The kernel function is polynomial with third order. Then the result passed through the yellow block responsible of the concatenation of Gramian matrix elements. Finally, the equation $y(x)=\sum_{i=1}^{Nsv}(\alpha_i^*\alpha_i)K(x_i,x)+b$ is ready and displayed.

As mentioned in Fig. 6 the regression vector is 2d-vector: $x(i)=(y(i-1),y(i-2))$ The computation of this type of vectors is generally difficult in hardware. Consequently, the streaming approach is used for this vector. The two columns of the regression vector are streamed in parallel and simultaneously computed by the kernel processing function and support vectors. The kernel function is polynomial with third order. Then the result passed through the yellow block responsible of the concatenation of Gramian matrix elements. Finally, the equation $y(x)=\sum_{i=1}^{Nsv}(\alpha_i^*\alpha_i)K(x_i,x)+b$ is ready and displayed.

After verifying the of the system functionality on the Simulink environment the generation of hardware components are executed. While building the hardware system, ISE flow generates a bit-stream that will be later used to configure the FPGA. When the compilation is finished, a new one block is created including all the purposes necessary for the executing system on FPGA. The produced hardware capsule the SVM testing phase is allied to a bit-stream file. After downloading this file in the FPGA via the Digilent USB JTAG Cable, then System Generator reads the output back from JTAG and sends it to Simulink. When execution is accomplished, the displayed results are compared to the results expected by the simulation.

The more important criteria in this comparison is the computation time because the software (MATLAB) is incapable to realize matrix multiplication for big dimension (more than 1000). Therefore, the use of FPGA gains a lot of time and big data to be computed in one time. In Table 3, we illustrate the results for computing the testing phase on hardware (FPGA) called SVMsoft and software (MATLAB) called SVMhard for polynomial and sigmoid kernel.

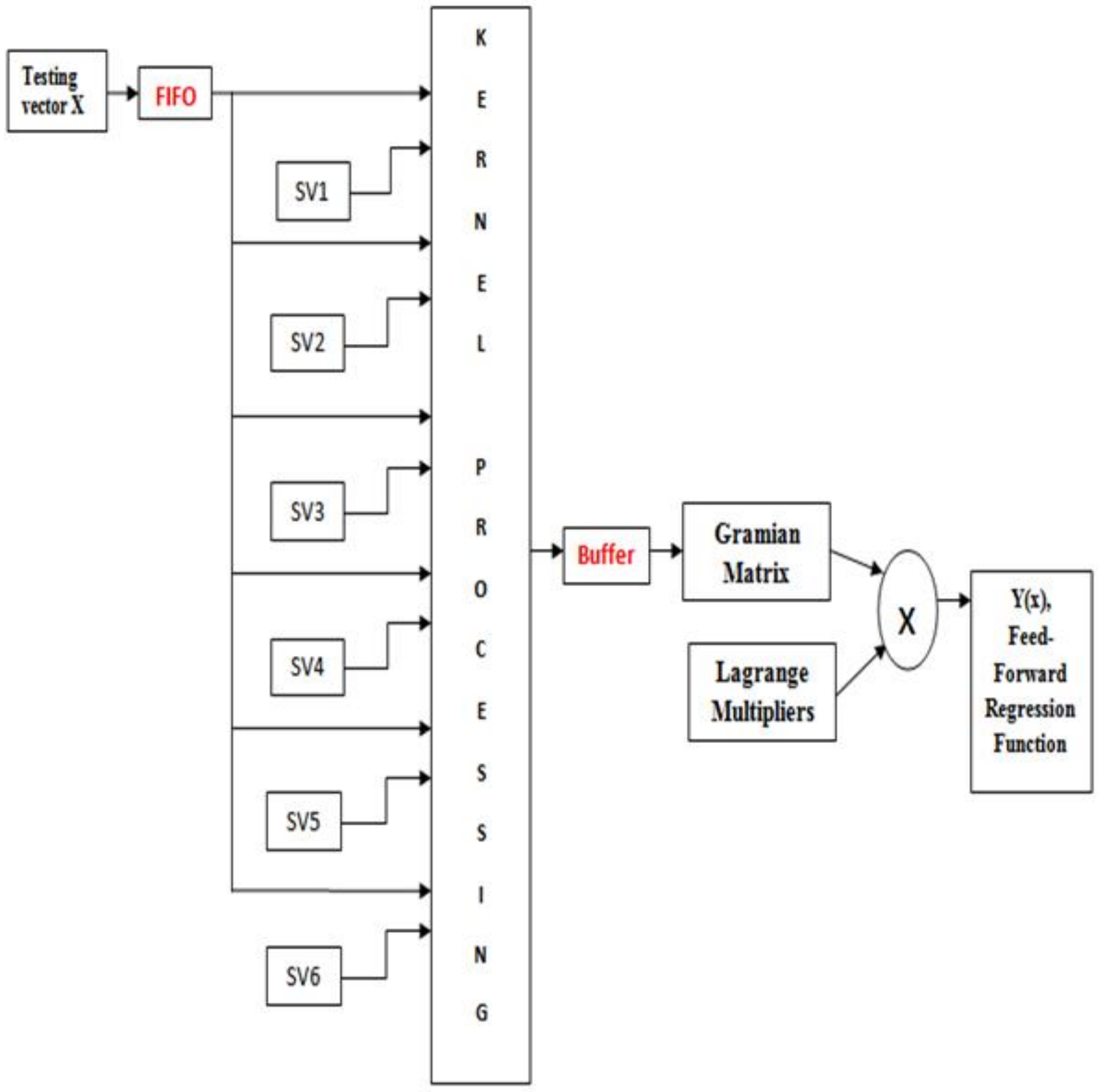


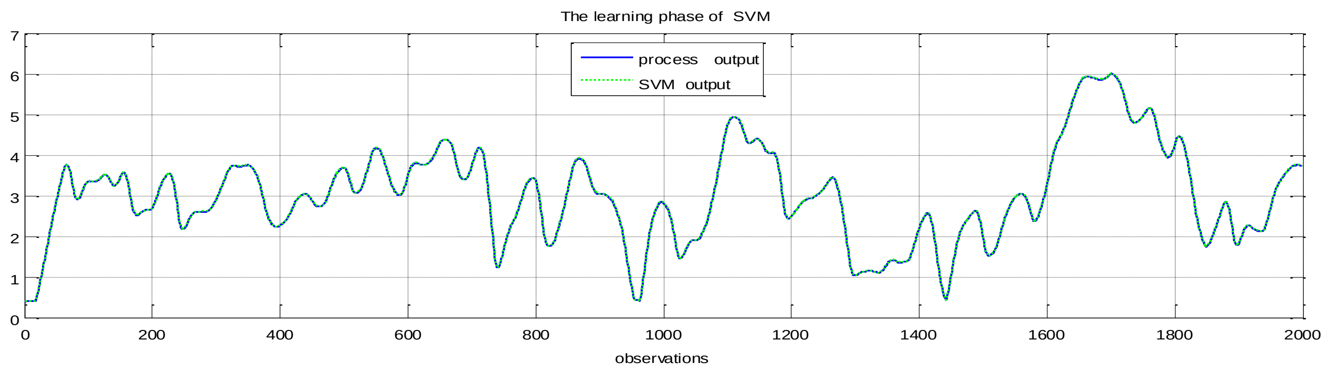
Fig. 6. Architecture for the feed-forward estimation function.

TABLE III. SVMSOFT AND SVMHARD PERFORMANCES

	Kernel type	NMSE Testing	CT(s)
SVMsoft	**Polynomial**	$5.0766.10^{-09}$	740.553
SVMhard		$9.0463.10^{-03}$	12.032044
SVMsoft	**Sigmoid**	$4.8928.10^{-08}$	664.931
SVMhard		$2.1974.10^{-02}$	10.056215

The exploited FPGA platform was VIRTEX 5 with the clock time period is 10 ns. The difference of computation time between SVMsoft and SVMhard was very big in order of 60 times. This acceleration was reached thanks to the FPGA computation power. The error rate of SVMhard calculated by the Normalized Mean Squared Error (NMSE) is higher than error rate of SVMsoft. The little difference in accuracy was in order 10^{-5} caused by the fixed point arithmetic used in hardware implementation.

We draw the different plots of SVMsoft, SVMhard and the real output of process in the same Fig. 7. As seen, there is a good concordance between the three plots that demonstrate the efficiency of the adopted approach.



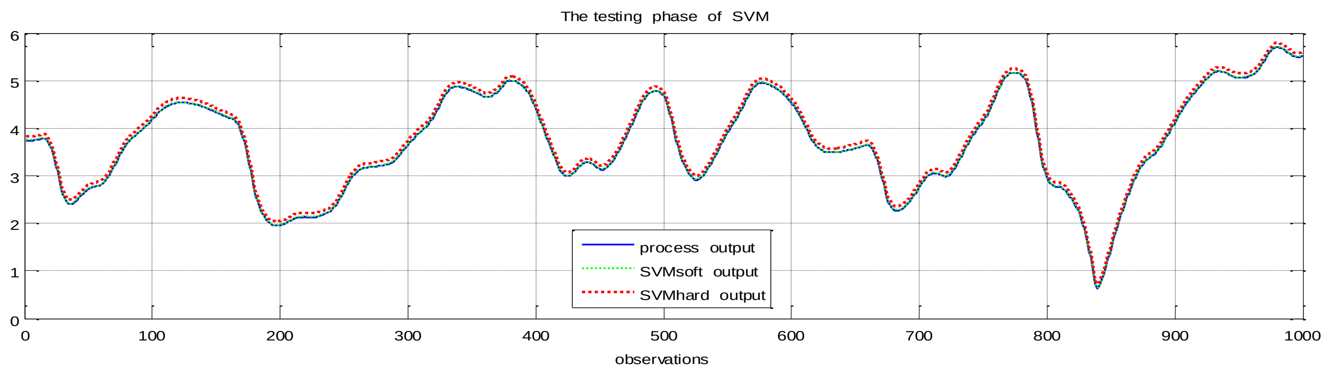


Fig. 7. Learning and testing phase of SVM (hard & soft).

VI. Comparison with Similar Work

In this section we propose to compare the performances of our design to the design of Marta Ruiz-Llata [6] that calculates the testing phase of SVM for classification and regression by the same design. As we interested to the regression task, we implemented the same system used in Marta work. It was the sinus cardinal function "sinc" corresponded to this expression:

$$y = \frac{\sin\sqrt{x_1^2 + x_2^2}}{\sqrt{x_1^2 + x_2^2}} \tag{10}$$

To obtain convincing model for testing the prediction function quality the selected datasets were arbitrarily engendered by 400 input (x_1, x_2) values appertaining to $x_i \in$ (10, 10) and conniving its corresponding output value y.

The training phase and parameters selection were also achieved by MATLAB. After finding the optimal values like the regularization parameter C, the insensitive zone ε and the parameter of kernel function γ, the efficiency of the model have to be tested with new unknown data. Then the estimated values were compared with the true values for sinus cardinal function. The testing dataset was composed of 100 arbitrarily values chose in the same way as testing dataset.

The author of this work employed the hardware friendly kernel function described by this expression:

$$2^{-\gamma\|x_i - x\|_1}$$

Where $\gamma = 2^{-2}$, C=1 and ε=0.01. There were 283 support vectors. The middling error between the predicted output and the real one was 0.02. The selected device for implementation was Altera EP2C20 Cyclone II using 8 bits resolution with fixed point arithmetic for representing data. The clock rate of the system was restricted to 30 MHz.

With the same method SVM for regression, the same approach (implementing just the testing phase) and the same platform (FPGA), we exploited our hardware architecture to implement the same sinus cardinal function for the same data sets. The platform exploited by Marta was very old. Therefore, there is no benefit to implement our design on Altera cyclone 2.

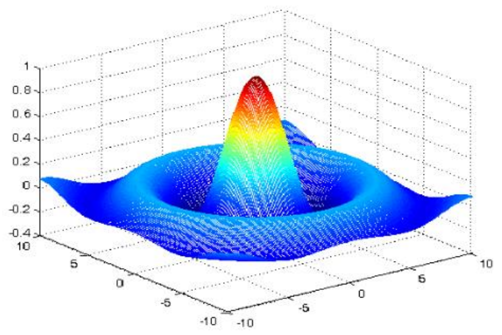


Fig. 8. A three dimensions plot of the sinus cardinal function.

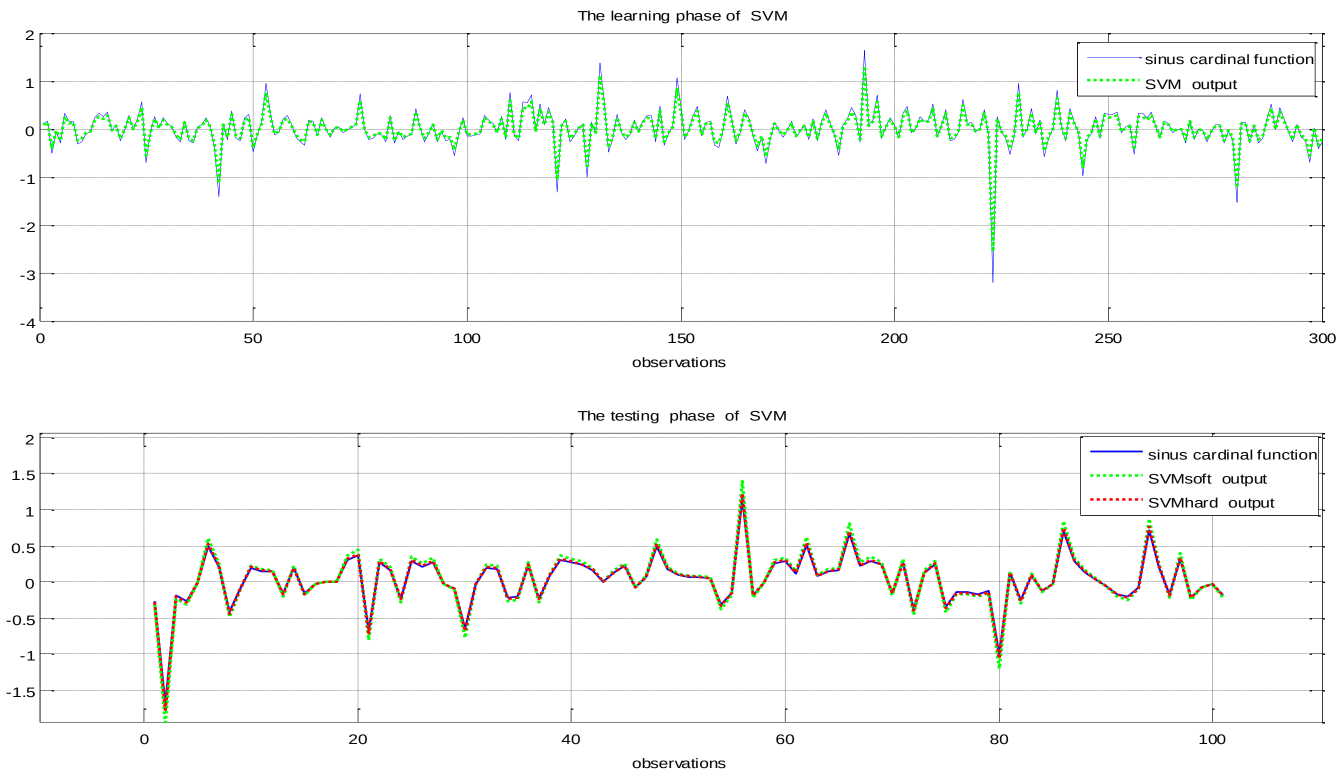


Fig. 9. Learning and testing phase of SVM for sinus cardinal function.

The used kernel function was the sigmoid function with the parameters: α=1 and c=10. The optimal parameters of SVM were C=100 and ε=0.001. There were just 10 support vectors. The reached performances were better in term of computation time and error rate. The NMSE was equal to $5.4437.10^{-04}$ and the total time was 08.1075 seconds. In Fig. 8, the different outputs are drawn. In the learning phase, the sinus cardinal function and SVM output are drawn. Then in the testing phase, also the sinus cardinal function is plotted with SVM output (SVMsoft) and the SVMhard (the implementation result).

In Fig. 9, it seems clear the strong resemblance between the sinus cardinal function output and the expected output in the training and learning phase. The SVMsoft and SVMhard were approximately confused thanks to the effectiveness of the SVM method.

Therefore, it is easy to predict the output of any process either linear or nonlinear in very short time. The number of support vectors is based on the value of the margin ε.

VII. Conclusion

In this paper, an efficient method of implementing the testing phase of SVM method was advised. The basic contribution of this approach is to accelerate the computation of the RKHS model by use of powerful FPGA. The experiments prove the excellent speedup attained that is more than 60 times compared with the software computation time.

The advantage of the designing tool Xilinx System Generator is the possibility to implement the software and the hardware in the same environment. Furthermore, Simulink offers a pleasant graphics interface for supple modellization and simulation. The design was well organized into streaming approach along the testing phase.

This practice permitted to construct a robust model for nonlinear system using novel data in the testing phase.

Future works will incorporate the use of the Xilinx System Generator for the development of other kernel functions to increase the simulation precision. Also, better performances can be reached with newer and more powerful FPGA type.

REFERENCES

[1] Lennart Ljung. Some aspects on nonlinear system identification. In Proc 14th IFAC Symposium on System Identification, Newcastle, Australia, March 2006.

[2] Lennart Ljung. Perspectives on system identification. Annual Reviews in Control, 34(1), March 2010.

[3] Alaa F. Sheta, "A Comparison between Regression, Artificial Neural Networks and Support Vector Machines for Predicting Stock Market Index ", communication on (IJARAI) International Journal of Advanced Research in Artificial Intelligence, Vol. 4, No.7, 2015

[4] Bernhard Scholkopf and Alexander J. Smola, "Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond", MIT Press Cambridge, MA, USA 2001

[5] Cristian Preda, "Regression models for functional data by reproducing kernel Hilbert spaces methods", Journal of Statistical Planning and Inference Volume 137, Issue 3, 1 March 2007, Pages 829–840.

[6] M.Ruiz-Llata, G. Guarnizo, and M. Yébenes-Calvino, "FPGA Implementation of a Support Vector Machine for Classification and Regression," in The 2010 International Joint Conference on Neural Networks (IJCNN), 2010, pp. 1-5.

[7] D. Anguita, S. Pischiutta, S. Ridella, and D. Sterpi, "Feed-Forward Support Vector Machine without Multipliers," IEEE Transactions on Neural Networks, vol. 17, pp. 1328-1331, 2006.

[8] X. Pan, H. Yang, L. Li, Z. Liu, and L. Hou, "FPGA Implementation of SVM Decision Function Based on Hardware-friendly Kernel," in International Conference on Computational and Information Sciences , ICCIS 2013 Proceedings, 2013, pp. 133-136.

[9] R. Patil, G. Gupta, V. Sahula, and A. Mandal, "Power Aware Hardware Prototyping of Multiclass SVM Classifier Through Reconfiguration," in 2012 25th International Conference on VLSI Design (VLSID), 2012, pp. 62-67

[10] C. Kyrkou and T. Theocharides, "A Parallel Hardware Architecture for Real-Time Object Detection with Support Vector Machines," IEEE Transactions on Computers, vol. 61, pp. 831-842, 2012.

[11] C.-W. Hsu and C.-J. Lin, "A Comparison of Methods for Multiclass Support Vector Machines," IEEE Transactions on Neural Networks, vol. 13, pp. 415-425, 2002

[12] B. Mandal, M. P. Sarma, and K. K. Sarma, "Implementation of Systolic Array Based SVM Classifier Using Multiplierless Kernel," in International Conference on Signal Processing and Integrated Networks (SPIN),2014,pp. 35-39.

[13] H. Hussin, K. Benkrid, and H. Seker, "Reconfiguration-Based Implementation of SVM Classifer on FPGA for Classifying Microarray Data," in 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2013, pp. 3058-306

[14] M. Wielgosz, E. Jamro, D. Zurek, and K. Wiatr, "FPGA Implementation The Selected Parts of Fast Image Segmentation," in Studies in Computational Intelligence vol. 390, ed, 2012, pp. 203-21

[15] M. Pietron, M. Wielgosz, D. Zurek, E. Jamro, and K. Wiatr, "Compariison of GPU And FPGA Implementation of SVM Algorithm for Fast Image Segmentation," in Architecture of Computing Systems–ARCS 2013, ed: Springer, 2013, pp. 292-302.

[16] Y. Ago, K. Nakano, and Y. Ito, "A Classification Procesor for Support Vector Machine with Embedded DSP Slice and Block RAM in the FPGA," in IEEE 7th International Symposium on Embedded Multicore Socs (MCSoC), 2013, pp. 91-96

[17] S. Kim, S. Lee, and K. Cho, "Design of High-Performance Unified Circuit for Linear and Non-Linear SVM Classifications," Journal of Semiconductor Technology and Science, vol. 12, pp. 162-167, 2012.

[18] J. Nayak, B. Naik, and H. Behera, "A Comprehensive Survey on Support Vector Machine in Data Mining Task: Applications & Challenges," International Journal of Database Theory and Application, vol. 8, pp. 169-186, 2015.

[19] J. Gimeno Sarciada, H. Lamel Rivera, and M. Jiménez, "CORDIC Algorithms for SVM FPGA Implementation," in Proceedings of SPIE - The International Society for Optical Engineering, 2010.

[20] H. Lamela, J. Gimeno, M. Jimenez, and M. Rruiz, "Performance Evaluation of FPGA Implementation of Digital Rotation Support Vector Machine," in SPIE Defense and Security Symposium, 2008, pp. 697908-697908-8

[21] T. Bollaert, "Catapult Synthesis: Practical Introduction to Interactive C Synthesis," SpringerLink Book Chapter, Pages 29-52, ISBN 978-1-4020-8587-1.

[22] Y. L.C.N.W. Wong.Generating hardware from OpenMP programs.Proc.IEEE Int.Conf. on Field Programmable Technology, pages73ñ80,Dec.2006.

[23] J. Gummaraju and M. Rosenblum.Stream Programming on General-Purpose Processors.In Proc.38th Int. Symp.on Micro architecture, pages343ñ354,Washington,DC,2005.