35th Annual International Conference of the IEEE EMBS
Osaka, Japan, 3 - 7 July, 2013

Reconfiguration-Based Implementation of SVM Classifier on FPGA for Classifying Microarray Data

Hanaa M. Hussain*, Khaled Benkrid and Huseyin Seker

Abstract— **Classifying Microarray data, which are of high dimensional nature, requires high computational power. Support Vector Machines-based classifier (SVM) is among the most common and successful classifiers used in the analysis of Microarray data but also requires high computational power due to its complex mathematical architecture. Implementing SVM on hardware exploits the parallelism available within the algorithm kernels to accelerate the classification of Microarray data. In this work, a flexible, dynamically and partially reconfigurable implementation of the SVM classifier on Field Programmable Gate Array (FPGA) is presented. The SVM architecture achieved up to 85x speed-up over equivalent general purpose processor (GPP) showing the capability of FPGAs in enhancing the performance of SVM-based analysis of Microarray data as well as future bioinformatics applications.**

Keywords— Bioinformatics, Field Programmable Gate Array, General Purpose Processor, Microarray, Support Vector Machine, Dynamic Partial Reconfiguration.

I. INTRODUCTION

Recent advances in biotechnologies have resulted in generation of overwhelming amount of high throughput biological data. Microarray, which is used to measure the expression profiles of tens of thousands genes simultaneously is one of the main contributors to the big data. Processing Microarray data is necessary as to extract the biological relevance embedded within the data, but this task is highly computational. Pre-processing Microarray data results in numeric matrices of gene expression profiles across variable samples which should further be analyzed by using supervised or unsupervised computational learning algorithms to transform the numeric matrices into medical knowledge. This type of Microarray data analysis has helped scientists in identifying genes associated with diseases or conditions (e.g., cancers), discovering drugs, personalizing treatment plans, and predicting treatment outcomes. In addition, more efforts are paid towards learning more about the regulation and interaction between genes to uncover new classes of tumours and to develop genomic based predictive models [1]-[5].

Hanaa M. Hussain, is with the Electronics Engineering Department, College of Technological Studies, The Public Authority of Applied Education and Training, Shuwaikh 70654, Kuwait (e-mail: hmh.hussain@paaet.edu.kw).

Khaled Benkrid, is with the School of Engineering and Electronics, The University of Edinburgh, Kings Buildings, Mayfield Road, Edinburgh EH9 3JL, U.K. (e-mail: k.benkrid@ed.ac.uk).

Huseyin Seker is with the Bio-Health Informatics Research Group, Centre for Computational Intelligence, De Montfort University, Leicester, LE1 9BH, U.K., (e-mail: hseker@dmu.ac.uk).

*Corresponding Author (hmh.hussain@paaet.edu.kw)

There have been different classifier architectures applied to Microarray data for prognostic and diagnostic decision making by evaluating a set of the genes that are found to have been associated with diseases or conditions [4]-[5]. Support Vector Machine-based classifier architecture (SVM) is one of the widely used classifiers in analyzing Microarray data and has been shown to perform better than those reported in the literature, mainly due to its capability of dealing with high dimensional data, flexibility in choosing a similarity function and ability to identify outliers [4]. In addition, SVM is characterized as having kernels that can be parallelized leading to increased performance [6]. Taking this concept into consideration, in this work, a Field Programmable Gate Array (FPGA) implementation of the SVM classifier is proposed and tested to assess the viability of FPGAs as efficient high performance solution in the analysis of bio-medical data, particularly in Microarray as a case study.

The rest of the paper is organized as follows; Section II will provide background on FPGAs and SVM, and Section III will summarize relevant works on the area. Section IV will then present the FPGA architecture of the SVM classifier. In section V, the implementation results will be presented. Finally, the conclusion and future work will be stated in Section VI.

II. BACKGROUND

A. FPGAs

FPGAs are reconfigurable computing platforms which have been evolving at a rapid pace over the last three decades, growing as Integrated Circuits (ICs) of few hundreds of logic gates to several millions, and integrating other heterogeneous resources within the IC. FPGAs are based on ICs containing enormous amount of small logic cells that can be configured or programmed to carry out many logical operations specified in the Hardware Description Language (HDL) code [7].

One of the main advantages of FPGAs is that they can be configured to execute multiple instructions in parallel and can pipeline tasks leading to high computing performance. The level of parallelism inherent in FPGAs is responsible for their popularity in applications requiring high performance, given that such applications lend themselves to hardware implementations. Today, FPGAs have been successfully used as accelerators to many applications serving as coprocessors to general purpose processors (GPPs) [8].

B. Support Vector Machine

The Support Vector Machine (SVM)-based classification is used to assign a class label to a new sample whose class

label is to be predicted by learning from a set of data with known class labels. SVM consists of two discreet phases; one is the training phase while the other is the evaluation of the decision function or the classification phase. During the training phase, SVM estimates a function which classifies the data into two classes by forming a hyperplane that maximizes the separation of the two classes [9]. SVM deals mainly with problems of binary classes (e.g., class label= 1 or class label= -1), and when multi-class problems are used, the SVM is performed on two classes' at a time until all the classes are covered.

Given a training set (x_i,y_i), where i=0 to N-1 (N is the number of training samples), $x_i \subseteq \Re^M$ are the training features, M is the number of features or dimensions, and $y_i \in \{-1,1\}$ being the known classifications of all the training samples. The classification function of linearly separable training data is given in (1) [10]:

$$f(x) = \langle w.x \rangle + b, \tag{1}$$

where b is the bias or the distance between the hyperplane and the origin, and w is a normal vector of the separating hyperplane. The hyperplane seeks the maximization of the distance between two soft margins through the minimization of the norm w for linearly separable training data. The minimization of w leads to applying a Lagrangian function given in (2) to optimize the solution [9]:

$$L(w,b,\alpha) = \frac{\|w\|^2}{2} - \sum_{i=0}^{N-1} \alpha_i \{[(x_i.w) - b] y_i - 1\}, \tag{2}$$

where α_i's are the Lagrange multipliers. L has to be minimized with respect to w and b; and maximized with respect to $\alpha_i > 0$. Based on the Kühn-Tucker theorem, which implies that for an optimum hyperplane, α_i must be ≥ 0, w can be expressed as in (3) for linear SVM.

$$w = \sum_{i=0}^{N-1} y_i \alpha_i x_i \tag{3}$$

Substituting (3) into (2) and applying the associated constrains leads to the dual formulation given in (4).

$$L(\alpha) = \sum_{i=0}^{N-1} \alpha_i - \frac{1}{2}\left[\sum_{i,j=0}^{N-1} \alpha_i \alpha_j y_i y_j K(x_i, x_j)\right], \tag{4}$$

$$given \sum_{i=0}^{N-1} \alpha_i y_i = 0, and \alpha_i \geq 0$$

The bias b is set to zero here assuming that the hyperplane is passing through the origin. K(.) in (4) is the kernel function which can be linear, Gaussian, or polynomial as given in 5a, 5b and 5c, respectively:

$$k(x_i, x_j) = x_i.x_j \tag{5a}$$

$$k(x_i, x_j) = e^{\left(\|x_i - x_j\|^2 / 2\sigma^2\right)} \tag{5b}$$

$$k(x_i, x_j) = (1 + x_i . x_j)^p. \tag{5c}$$

The linear SVM classifier is considered for the hardware implementation proposed in this paper; consequently the linear kernel in (5a) is used leading to the transformation of the classification function in (1) to (6):

$$f(x) = \sum_{i=0}^{N-1} y_i \alpha_i K(x_i.x_j) + b. \tag{6}$$

During the classification phase, the SVM classifier performs the operation in (7a) to determine in what side of the hyperplane the query vector Q lies as outlined in (7b) and (7c), respectively [9]:

$$\text{Query Class(Q)} = \text{sgn}\left(\sum_{i=0}^{N-1} y_i \alpha_i x_i^T Q\right), \tag{7a}$$

$$\text{Query Class(Q)} \geq 0 \Rightarrow Q \in C_1, \tag{7b}$$

$$\text{Query Class(Q)} < 0 \Rightarrow Q \in C_{-1}, \tag{7c}$$

where C_1 and C_{-1} are the binary class labels associated with y_i. In this work, the proposed FPGA implementation determines (7) for each query given that the training phase is done off-line and the hardware design is supplied with support vectors (SVs) having non-zero coefficients.

III. Relevant previous works

Most contributions on the FPGA implementation of the SVM classification have mainly dealt with non-biomedical data. The earliest work reported in the literature was in [6] where the authors proposed and implemented a digital architecture of SVM in FPGA targeting the training phase. The work did not include acceleration results with respect to GPP and it was mainly focused on proving the suitability of the application to hardware implementation. The same authors have various subsequent works, one was reported in [11] where they presented FPGA core generator tool for automatically generating Gaussian kernel SVM architecture in VHDL based on user requirements entered using graphical user interface (GUI) [11].

In [12], the authors reported SVM architecture that performs the training phase based on Sequential Minimal Optimization (SMO). The main contribution of the work was to implement the SMO-SVM using DPR, whereby the modular blocks performing the tasks associated with SVM training were time multiplexed leading to saving in the area occupied by the SVM core within the FPGA.

In [13], the authors reported a hardware implementation of the SVM classifier which performs both training and classification on FPGA based on three kernels: linear, Gaussian, and polynomial. The architecture targeted disease diagnosis based on using Microarray data which is closely related to the work presented here. The authors achieved superior performance in terms of classification, however they did not compare the FPGA implementation with GPP. In addition, a main drawback of their SVM architecture was its area footprint

In [14], the authors presented the FPGA implementation of SVM classification targeting brain computer interface assuming the training was performed off-line. When comparing the FPGA implementation with an equivalent GPP implementation, the FPGA performed worse in terms of processing speed consuming twice more time than GPP. In addition, the FPGA architecture was non-scalable and limited to six dimensions only [14].

IV. Architecture Design of the Proposed Model

The proposed SVM architecture is a modular systolic array consisting of four blocks as shown in Fig. 1, captured in Verilog HDL to compute (6) and determine the sign in (7).

The first block is the memory, which is responsible for storing the data (i.e., training and query data) and broadcasting them to the second block. The second block is the kernel computation block, the third and fourth blocks are the accumulation and decision making blocks, respectively which all will be described below.

The kernel computation block is partitioned into three sub-blocks operating in two stages as shown in Fig. 2, which are pipelined to perform portion of the computation. The first sub-block consists of a systolic array of a number of SV kernel PEs as detailed in Fig. 3(a) where each PE has the role of receiving one SV feature every clock cycle (x_{ij}) along with the corresponding query feature (Q_j). One feature of the query FIFO is read by the first kernel processing element (PE) every clock cycle and propagated through the pipeline allowing for parallel SV kernel computations as shown in Fig. 3(a). In the first stage, the systolic array is known as Multiplier A to basically compute (8):

$$\text{Multiplier A} = \sum_{j=0}^{M-1} x_j Q_j, \tag{8}$$

where x_j is a support vector (SV) feature, Q_j is the corresponding query feature. The systolic array is fully parallelized such that all the PEs work simultaneously to compute (8). This operation is mainly facilitated by the capability to obtain the needed feedstock for each PE continuously from the local memory attached to each PE and propagated through the pipeline as shown in Fig. 3(a). The functionality of each PE is illustrated in Fig. 3(b). The latency of the pipeline is M clock cycles, while the throughput is one result per clock cycle. Consequently, for processing one query vector, M+SV clock cycles are required by the pipeline to finish the computation. Just after a period of M-1 clock cycles, the second sub-block in the kernel computation block known as Multiplier B starts reading the training coefficients (α_i) and the class labels (y_i)

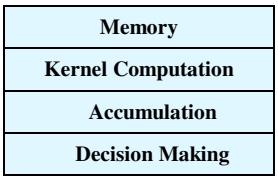

Figure 1. The modular blocks of the proposed SVM architecture.

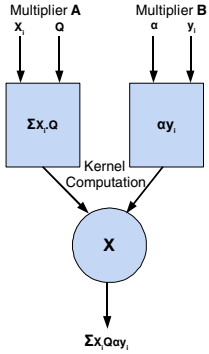

Figure 2. Datapath of the kernel computation block.

associated with each SV from the Memory block to compute the scalar product shown in (9):

$$\text{Multiplier B} = \alpha_i y_i . \tag{9}$$

To be able to complete the computation of the kernel for a single SV, the third sub-block shown in stage two of Fig. 2 combines the results of multipliers A and B to obtain the final result as given in (10):

$$\text{Kernel Computation} = \sum_{j=0}^{M-1} X_{ij} Q_j \alpha_i y_i . \tag{10}$$

The third block of the SVM core is a simple add-and-accumulate circuitry to accumulate the results as they come in from the kernel computation block.

Finally, the fourth block known as the decision making block determines the class label of the query based on the sign of the accumulation result obtained, where class zero distinguishes a diseased tissue and class one is a healthy one. The aforementioned four blocks form together the complete SVM core.

The DPR implementation of the SVM classifier is constructed using Xilinx' PlanAhead 12.2. tool to create various copies of the complete SVM core based on different parameters (i.e., number of SVs, features, coefficients, and wordlengths) which can be used to reconfigure the FPGA during run-time. This DPR feature allows for swapping a complete SVM core that is already placed on the FPGA with another one while the device is running without interfering with the operation of other tasks placed elsewhere on the device.

V. Implementation Results

The hardware implementation was tested on FPGA platform board, namely, Xilinx ML 403 using synthetic Microarray data of size that can be stored within the Block RAMs of the available FPGA device. On the other hand, the software implementation on GPP was based on Matlab (R2009b) bioinformatics toolbox running on a 2.60 GHz Pentium Dual-Core E5300, with 3 GB RAM workstation. The toolbox includes an optimized SVM classification function that can be easily utilized. The SVM core was simulated first, then synthesized, mapped, placed and routed using Xilinx ISE 12.2 to target the *XC4VFX12* FPGA available on board Xilinx ML 403 platform board [15]-[16].

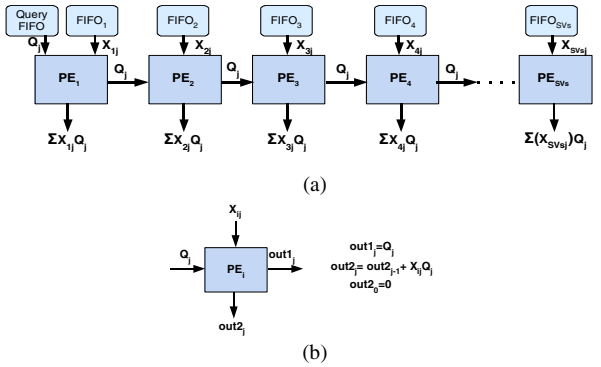

Figure 3. (a) The systolic array of Multiplier A where PEs perfom multiplations in parallel. (b) The functionality of a single kernel PE.

The implemented design was based on the parameters; B=8, M=1024, and SVs=20.The FPGA was first configured using JTAG cable and then run. The implementation was then tested using Xilinx' ChipScope™ Pro Analyzer 12.2 and checked against simulation results. The number of clock cycles to classify one query was found to be 1048 cycles. Table I summarizes the performance results of the FPGA and GPP implementations, it shows that the FPGA implementation outperformed the GPP implementation by ~61 times. As for the area footprint, the SVM core occupied 31% of the device area.

The same design was also implemented using a higher end FPGA, namely Xilinx' *XC4VSX35*, achieving the results shown in Table 1 whereby the hardware design attained a speed-up of ~85 times over an equivalent GPP implementation, this finding was based on simulation results only due to the unavailability of this FPGA device.

As for the DPR implementation of the SVM classifier, the full reconfiguration time required to place one SVM core onto the FPGA chip was 202.78 ms based on using JTAG cable as configuration port. On the other hand, the partial reconfiguration time to replace the SVM core already running on FPGA with a variant SVM core having different parameters was 24.12 ms. As such, partially reconfiguring the FPGA was found to be ~8x quicker than reconfiguring the whole FPGA while maintaining the operation of other tasks placed on the same FPGA. The latter is particularly crucial when multiple users are sharing the FPGA since reconfiguring the whole FPGA will interrupt their tasks.

VI. Conclusion and Future Work

The proposed hardware implementation of the SVM classifier on FPGA realizes high performance customized solution applied to Microarray data analysis, which outperforms GPPs in terms of execution. The FPGA implementation of the SVM classifier is up to ~85 times quicker than an equivalent implementation running on GPP. Furthermore, the proposed implementation is adaptive to user requirements. As for the DPR implementation, it was found that partially reconfiguring the FPGA is ~8x faster than full device reconfiguration. This means that changing parameters in the SVM core can be performed quickly while the device is running without interrupting other tasks. Thus,

TABLE I. Summary of Timing Performance of the SVM Core

FPGA Device	Clock Speed (MHz)	GPP Software (µs)	FPGA (µs)	Speed- up
XC4VFX12	98.7	646	10.62	~61
XC4VSX35	137.7	646	7.64	~85

it can be stated that FPGAs provide high performance solution for the analysis of Microarray data, and could be applied to process other bio-medical data requiring high computational power as a result of the continuous growth in data throughputs. Future work will focus on implementing SVM training on FPGA using DPR, testing with benchmark datasets on state-of-the-art FPGAs, and implementing the classification functions using different kernels. Furthermore, FPGA architectures of the ensemble SVM classifier will be implemented using DPR.

References

[1] D. Stekel, *Microarray Bioinformatics*. Cambridge, U.K: Cambridge Univ. Press, 2003.

[2] K. Le Cao and G. McLachlan, "Statistical Analysis on Microarray Data: Selection of Gene Prognosis Signatures," in *Computational Biology: Issues and Application in Oncology* (Applied Bioinformatics and Biostatistics in Cancer Research series), 1st ed., T. Pham Ed. New York : Springer, 2009, ch. 3, pp. 55–75.

[3] A.K. Jain, M.N. Murty, and P.J. Flynn, "Data Clustering: A Review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

[4] M. P. S Brown *et al.*, "Knowledge-Based Analysis of Microarray Gene Expression Data Using Support Vector Machines," in *Proc. Natl. Acad. Sci (PNAS)*, vol. 97, no. 1, pp. 262–267, Jan. 4, 2000.

[5] S. Mukherjee, "Classifying Microarray Data Using Support Vector Machines," in *A Practical Approach to Microarray Data Analysis*, 1st ed., D. Berrar *et al.*, Eds. USA: Springer, 2009, ch. 9, pp. 1–19.

[6] D. Anguita, "A Digital Architecture for Support Vector Machines: Theory, Algorithm, and FPGA Implementation," *IEEE Trans. Neural Networks*, vol. 12, no. 5, pp. 993–1009, Sep. 2003.

[7] D. Buell, T. El-Ghazawi, K. Gaj, and V. Kindratenko, "High Performance Reconfigurable Computing," *IEEE Computer*, vol. 40, no. 3, pp. 23–27, Mar. 2007.

[8] T. El-Ghazawi *et al.*, "The Promise of High-performance Reconfigurable Computing," *IEEE Computer*, vol. 41, no. 2, pp. 69–76, Feb. 2008.

[9] V. Vapnik, *The Nature of Statistical Learning Theory*. New York, US: Springer-Verlag, 2000.

[10] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and other Kernel-Based Learning Methods*. USA: Cambridge University Press, 2000.

[11] D. Anguita, L. Carlino, A. Ghio, and S. Ridella, "A FPGA Core Generator for Embedded Classification Systems," *J. Circuits, Systems, and Computers*, vol. 20 , no. 2 , pp. 263–282, Apr. 2011.

[12] J. Gomes Filho, M. Raffo, M. Strum, and W. Jiangg Chau, "A General-purpose Dynamically Reconfigurable SVM," in *Proc. 6th Southern Programmable Logic Conference (SPL)*, Ipojuca, Brazil, Mar. 24–26, 2010, pp. 107–112.

[13] J. Woo Wee and C. Ho Lee, "Concurrent Support Vector Machine Processor for Disease Diagnosis," in *Neural Information Processing (Lecture Notes in Computer Science)*, Pal *et al.*, Eds., Berlin Heidelberg: Springer-Verlag, 2004, vol. 3316, pp. 1129–34.

[14] O. Pina-Ramirez, R. Valdes-Cristerna, and O. Yanez-Suarez, "An FPGA Implementation of Linear Kernel Support Vector Machines," in *Proc. Int. Conf. on Reconfigurable Computing and FPGAs*, San Luis Potosi, Mexico, Sep. 27–29, 2006, pp. 1–6.

[15] Xilinx Inc., "Xilinx ML401/ML402/ML403 Evaluation Platform User Guide ug080," v. 2.5, 2006. [Online at http://www.xilinx.com. accessed April 20, 2012].

[16] Xilinx Inc., "Virtex-4 User Guide ug070", v. 2.6, 2008. [Online at http: //www.xilinx.com. accessed April 20, 2012].