Dynamic Partial Reconfiguration implementation of the SVM/KNN multi-classifier on FPGA for Bioinformati....

Conference Paper · August 2015

DOI: 10.1109/EMBC.2015.7320168

CITATIONS
0

READS
102

3 authors, including:

Hanaa Mohammad Hussain
Public Authority for Applied Education and Trai...
13 PUBLICATIONS **124** CITATIONS
SEE PROFILE

Huseyin Seker
Northumbria University
125 PUBLICATIONS **530** CITATIONS
SEE PROFILE

Dynamic Partial Reconfiguration implementation of the SVM/KNN multi-classifier on FPGA for Bioinformatics application

Hanaa M. Hussain, Khaled Benkrid, and Huseyin Seker

Abstract— Bioinformatics data tend to be highly dimensional in nature thus impose significant computational demands. To resolve limitations of conventional computing methods, several alternative high performance computing solutions have been proposed by scientists such as Graphical Processing Units (GPUs) and Field Programmable Gate Arrays (FPGAs). The latter have shown to be efficient and high in performance. In recent years, FPGAs have been benefiting from dynamic partial reconfiguration (DPR) feature for adding flexibility to alter specific regions within the chip. This work proposes combing the use of FPGAs and DPR to build a dynamic multi-classifier architecture that can be used in processing bioinformatics data. In bioinformatics, applying different classification algorithms to the same dataset is desirable in order to obtain comparable, more reliable and consensus decision, but it can consume long time when performed on conventional PC. The DPR implementation of two common classifiers, namely support vector machines (SVMs) and K-nearest neighbor (KNN) are combined together to form a multi-classifier FPGA architecture which can utilize specific region of the FPGA to work as either SVM or KNN classifier. This multi-classifier DPR implementation achieved at least ~8x reduction in reconfiguration time over the single non-DPR classifier implementation, and occupied less space and hardware resources than having both classifiers. The proposed architecture can be extended to work as an ensemble classifier.

I. Introduction

In bioinformatics and computational biology (BCB) applications, scientists tend to experiment with different supervised learning algorithms in order to determine the most suitable algorithm for the problem in hand and need to compare results obtained using different supervised algorithms. In some cases they combine results obtained from different methods in order to help determine best algorithms and more robust set of values of the algorithmic parameters. However, one of the obstacles of applying multiple supervised methods to large datasets associated with such big data sets is that using general purpose processors (GPPs) takes long execution times, and consumes high electrical power. One of the high performance solutions to overcome such limitations is the use of FPGAs.

In [1]-[4], the authors have proposed several FPGA architectures for two commonly used supervised algorithms in analyzing BCB data, namely, the K-nearest neighbor (KNN) classifier, and support vector machines (SVMs). The two FPGA architectures achieved high execution times compared to GPPs. Furthermore, the authors applied dynamic partial reconfiguration (DPR) to the aforementioned classifiers and found that it offers significant time saving. DPR basically allows the designer to create an adaptive and flexible system to modify a block of logic dynamically [5]-[6]. This capability is useful for introducing alternations to tasks running on the FPGA without re-powering the device or interrupting other tasks located in other regions of the FPGA. In our previous works presented in [1]-[4], the DPR implementations of the SVM and KNN classifiers saved at least ~8x in reconfiguration time compared to full chip configuration.

In this work, we propose the use of DPR to develop a multi-classifier which allows for interchanging various copies of the KNN and SVM cores. This architecture will enable users to configure a specific region on the FPGA to work as either KNN or SVM classifier. In addition, it could be further modified to combine the outcome of both classifiers to form an ensemble classifier on FPGA.

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

II. Background

A. Dynamic Partial Reconfiguration (DPR) of Xilinx FPGAs

FPGAs are heterogeneous Integrated Circuits (ICs) containing enormous amount of logic cells and dedicated circuits that can be configured to do many logical operations specified in hardware description language. They form a high performance computing platform due to their capability to execute many instructions in parallel. FPGAs have been adopted by many industries due to the level of parallelism inherent in them, unlimited re-configurability at low cost, the high density of logic gates embedded into them, relatively

This work was supported by the Public Authority of Applied Education and Training (PAAET) in Kuwait to sponsor the PhD study of Hanaa M. Hussain at Edinburgh University.

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

Khaled Benkrid is with School of Engineering & Electronics, Edinburgh University, Kings Buildings, Mayfield Road, Edinburgh EH9 3JL, U.K. (email: k.benkrid@ieee.org)

Huseyin Seker is with Department of Computer Science and Digital Technologies, Faculty of Engineering and Environment, University of Northumbria at Newcastle, U.K.
(e-mail: huseyin.seker@northumbria.ac.uk).

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

low power consumption, and immediate time to market. FPGAs have been serving as coprocessors to general purpose processors (GPPs), whereby specific segments of an algorithm running on a GPP can be ported to FPGA where they can be executed at much faster rate [7]. The process of mapping an algorithm onto the FPGA starts by describing the functionality using hardware description language (HDL), simulating the design, synthesizing, implementing, and verifying the design which all done using the FPGA's vendor software and tools. At last, a full bitstream file is generated by the tool which is downloaded to the FPGA to configure the logic cells [8].

Most applications of FPGAs have been based on static configurations involving downloading the full bitstream to the FPGA, as a result altering the functionality of the hardware involves modifying the HDL code, re-implementing the design, and generating a new bitstream to reconfigure the FPGA, and this requires re-booting the FPGA. Recent evolution in FPGAs has led to the possibility of modifying the configuration of the FPGA fully or partially without the requirement of system re-boot thus some tasks can remain unaffected. Dynamic partial reconfiguration (DPR) is a design flow which enables changing subset of the FPGA configuration while the device is operating. This feature is offered by some of the current FPGA vendors such as Xilinx [5].

The first step in the DPR design is to implement modular hierarchical design, which is based on having a top design consisting of several partitions. Secondly, a bottom-up synthesis is applied to create separate multiple netlists for each of the submodules. Third, Xilinx PlanAhead tool is invoked, and a new PR project is created targeting the same FPGA used when creating the netlists. This process involves importing the netlist of the top-design, define reconfigurable partitions (RPs) which are basically the reconfigurable blocks, and importing the various netlists created for each partition constituting the logics of the RP regions, those are called reconfigurable modules (RMs). The last stage is to create configurations which will finally result into two bitstream files, one is partial while the other is full which are used to configure the FPGA fully or partially [5], [8].

B. K-nearest neighbor classifier (KNN)

The K-nearest neighbor (KNN) classifier is one of the common supervised learning methods in BCB applications [9], [10]. KNN is based on using training data with known class label to classify a query vector to a class label. Training data are basically N x M matrix where N is the training samples and M is the number of features, with an additional column representing the class label of each training sample. The way the classifier works is by first calculating the distances between the query and all of the samples using a particular distance metric such as Euclidian or Manhattan. The SVM/KNN multi-classifier is based on a KNN architecture developed with the Manhattan distance as formulated in (1) [1]-[2]:

$$D(x_i, y_i) = \sum_{i=1}^{M} (| x_{Ni} - y_i |), \quad (1)$$

where x_{Ni} are the training samples and y_i is the query. At the end of the distance computation, the accumulative distances between the query and all training samples are compared and sorted according to the number of neighbourhoods (K). The classifier then determines the K-minimum distances and sorts them in a descending order along with the class labels associated with the corresponding samples. These K-minimum distances are known as the K nearest neighbors or KNNs. Finally, the classifier performs a voting on those KNNs to assign the query to the most encountered KNN. The distance computation is the most time consuming part within the algorithm since it is repeated many times. Therefore, FPGAs can be used to accelerate the KNN through parallelizing the distance computation part, and pipelining all the other operations [1], [2].

C. Support Vector Machine classifier (SVM)

SVM is one of the popular supervised learning methods used in BCB applications such as in gene expression for classifying samples at the molecular level, identifying genes' functions, diagnosing and predicting diseased tissues [11], [12]. SVM has been superior in performance to other types of classifiers mainly due to its ability to manage data with high dimensionality [13], [14]. SVM is based on projecting data from an input space to a large feature space and then separating features of two class labels using a hyperplane that is constructed with a kernel function. The SVM classification consists of two phases; one is the training phase while the other is the evaluation of the decision function known as the classification phase [13]. The implementation of the SVM/KNN multi-classifier is based on SVM core implementing the classification phase only and assumes training is performed offline and training data made available to the core. Therefore, details will not be given here about the training phase. During the classification phase, the training samples obtained from the training phase (now called support vectors (SVs)) are used along with Lagrange multipliers α's to classify a query vector Q and assign a class label to it by solving the classification function based on linear kernel shown in (2).

$$Query\ \ Class\ (Q) = \mathrm{sgn}\left(\sum_{i=0}^{N} y_i \alpha_i x_i^T Q \right), \quad (2)$$

$$Query\ \ Class\ (Q) \geq 0 \Rightarrow Q \in C_1,$$

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

where C1 and C-1 are the class labels associated with each of the known classifications, Q is the query vector, and α is the Lagrange multiplier obtained during the training phase. The data set is (x_i,y_i), where i=1 to N, $x_i \subseteq \Re^M$ are the training features, M is the number of features or dimensions, and $yi \in \{-1,1\}$.

III. THE DPR IMPLEMENTATION OF THE SVM/KNN CLASSIFIER

The KNN and SVM cores used in this SVM/KNN multi-classifier are based on the cores presented in [1]-[4]. Fig. 1 summarizes the main building blocks which constitute the SVM and KNN FPGA cores that we have developed.

The procedure we used to construct the SVM/KNN classifier on FPGA starts by creating a design wrapper which instantiates a black-box. The black-box is basically an empty core having the same I/O ports as the SVM classifier. The black box needs to be compatible in the number of I/O ports with both cores, therefore the SVM I/O ports are selected since they already include all the KNN I/O's plus an additional port that is exclusive to the SVM core as shown in Fig. 2. The black-box needs to be set as the reconfigurable partition (RP) which can be filled by the logic contents of either the SVM or KNN classifier. Following the definition of the RP region on the FPGA and determining its size based on the largest logic resource requirement, the next step is to create Reconfigurable Modules (RMs). RMs are several images of the KNN and SVM cores having various parameters or data sizes. For example, the performance of the KNN classifier is affected by the chosen number of neighbourhoods (the parameter K) [15], and that a classification problem may need to be repeated for different K's to ensure accurate classification. Several RM's have been generated to construct a collection of RM's that can be used to reconfigure the RP region. Following the implementation of several configurations, the bitstreams were generated to create a library of configuration files that can be used to configure the FPGA as shown in Fig. 3. This SVM/KNN implementation is unique compared to the DPR implementations of individual SVM and KNN cores reported in [1]-[4] in that the configuration of the RP region can be swapped with a totally different classifier core rather than a variable image of the same core.

IV. IMPLEMENTATION RESULTS AND DISCUSSION

The design was created and implemented using Xilinx ISE 12.2. The hardware implementations targeted Xilinx' *XC4VSX35* FPGA. The results were based on simulations, and the configuration times were estimated based on (3) [6]:

$$\text{Configuration Time} = \frac{\text{Size of the bitstream file}}{\text{Bandwidth of the Configuration Mode}} \quad (3)$$

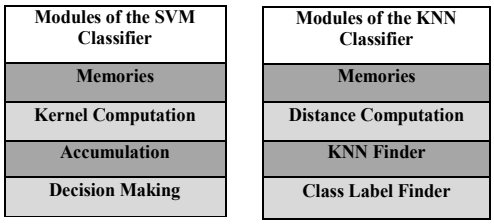
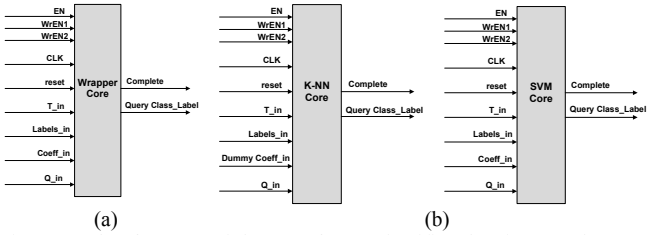


Figure 1. Block diagrams showing the modules for the SVM and KNN cores used in the FPGA implementation.



(a) (b)

Figure 2. The I/O ports of the SVM/KNN classifier showing: (a) the ports of the wrapper module is a black box that can be configured as either an SVM or KNN classifier, and (b) illustrates the I/O ports of both classifiers.

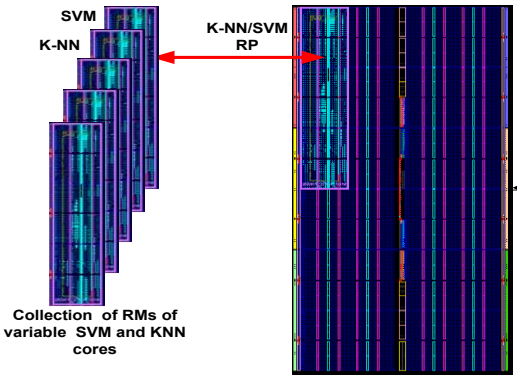


Figure 3. Floorplan image of the FPGA generated by the Xilinx PlanAhead tool highlighting the RP region and logic resources associated with it. The figure also illustrate the process of swapping the RP region with variants of the SVM and KNN classifiers (RM's).

where JTAG was selected as the configuration mode having a bandwidth of 66 Mbps [6]. The parameters used in the implementation are the same as those required by the two classifiers. They are: B=8, N=1024, SVs=1024, M=20, k=13, where B is the wordlength of the data used. The size of the RP region was set according to these parameters, leading to full and partial reconfiguration bitstream files of 1,673 and 199 KB in size, respectively. Table 1. summarizes the timing performance for initially configuring the FPGA with static or dynamic classifiers, and compares the partial reconfiguration timing for all three architectures. Table 1. shows that the dynamic SVM/KNN partial reconfiguration (DPR) is eight times faster than static partial configurations.

As for the area requirement of the proposed DPR SVM/KNN core, Table. 2 reveals that it occupies twice as less area on FPGA as an architecture having both SVM and KNN placed onto the same chip. In fact, our proposed SVM/KNN architecture occupies an area equivalent to a single static or dynamic SVM or KNN classifiers, yet have the capability to run any of the two classifiers in sequence. This implementation is useful for classifying big data sets with different classifiers.

TABLE I. COMPARATIVE RESULTS BETWEEN DIFFERENT CLASSIFIER ARCHITECTURES IN TERMS OF CONFIGURATION TIMES

FPGA architecture	Initial full configuration time (ms)	change the classifier type	Partial reconfiguration times	Speed-up
Static SVM	202.78	SVM to KNN	202.78	1x
Static KNN	202.78	KNN to SVM	202.78	1x
Dynamic SVM/KNN	202.78	SVM to KNN or KNN to SVM	24.12	8x

TABLE II. COMPARATIVE RESULTS BETWEEN DIFFERENT CLASSIFIER ARCCHITECTURES IN TERMS OF AREA FOOTPRINT BASED ON XILINX' *XC4VSX35* FPGA

FPGA architecture	CLB slices	Area requirement compared to KNN core
Dynamic or static SVM only	1442	~1x
Dynamic or static KNN only	1475	~1x
Static SVM and KNN placed together at the same FPGA	2917	~2x
Proposed DPR SVM/KNN core	1475	~1x

V. APPLICATION OF THE SVM/KNN IN BIOMEDICAL ENGINEERING

In BCB, supervised learning methods which include SVM, KNN, and neural networks are widely used in analyzing biomedical data such as Microarray data. Applying data mining to these data has led to the identification of some genes associated with specific diseases such as cancer [9-12]. Some of these discoveries were used to develop predictive models, which in turn led to the development of some commercial diagnostic kits used to diagnose and predict disease outcome. Examples of such kits are the MammaPrint, and Symphony developed by Agendia. Applying high performance computing such as FPGAs to data mine big biomedical data and form predictive models can lead to development of more diagnostic tests, and pave the way toward widely used personalized medicine. In addition, our hardware architecture can be used to develop ensemble classifiers for Microarray and other biomedical data to improve the accuracy of the prediction models. Such applications suffer from very long execution times and high energy consumption when done on GPP's which deter scientists from tackling complex problems. Although we are targeting BCB data, this architecture can be adapted by other applications that have similar requirements.

VI. CONCLUSIONS AND FUTURE WORKS

In this work, a DPR FPGA implementation of the SVM/KNN multi-classifier was presented for analyzing BCB big data. The proposed architecture allows a defined region within the FPGA to be dynamically adaptive to user's requirements for the SVM or KNN classifiers. This solution is an alternative to having to configure the same FPGA with two classifiers, therefore, it saves in reconfiguration time, area footprint, and in power consumption. The implementation has several advantages, first it offers time saving when modifying the defined reconfigurable regions to run another classifier, whereby ~8x speed up in reconfiguration time was achieved. Secondly, reconfiguring particular region within the FPGA consumes much less power than having to reconfigure the whole FPGA. Third, this implementation reduces space and area footprint occupied by the hardware design, and allows the use of smaller FPGAs for running multiple tasks instead of having to purchase large FPGAs to host multiple tasks. The latter also reduces the cost of the device.

Given the promising outcomes, future work will include testing the proposed architecture on a real FPGA to verify the aforementioned results using benchmark BCB data. In addition, we consider applying time-multiplexed ensemble classifier to run each of these two classifiers in sequence and then to combines the results of the two classifiers as this could lead to a more accurate and reliable decision.

ACKNOWLEDGMENT

Special thanks to the "Public Authority of Applied Education and Training for funding the research (PAAET)". We would also like to thank "Kuwait Foundation for Advancement in Science (KFAS)" for its support.

REFERENCES

[1] H. Hussain, K. Benkrid, and H. Seker, "An Adaptive Implementation of a Dynamically Reconfigurable K-Nearest Neighbour Classifier on FPGA," *in Proc. NASA/ESA Conf. on Adaptive Hardware and Systems (AHS)*, Nuremberg, Germany, June 25-28, 2012, pp.205-212.

[2] H. Hussain, K. Benkrid, C. Hong, and H. Seker, "An Adaptive FPGA Implementation of Multi-core K-Nearest Neighbour Ensemble Classifier Using Dynamic Partial Reconfiguration," *in Proc. Int. Conf. on Field Programmable Logic and Applications (FPL12)*, Oslo, Norway, Aug 29-31, 2012, pp. 627-630.

[3] H. Hussain, K. Benkrid, and H. Seker, "Dynamic partial reconfiguration based implementation of svm classifier for classifying microarray Data," *in proc. 35th Annu. Int. Conf. on IEEE Engineering in Medicine and Biology Society*, Osaka, Japan, 3-7 July 2013, pp. 3058-3061.

[4] H. Hussain, K. Benkrid, and H. Seker, "Novel dynamic partial reconfiguration implementations of the support vector machines classifier on FPGA", *Turk J. of Electr Eng Co*, to be published.

[5] D. Dye, "Partial Reconfiguration of Virtex FPGA in ISE 12," Xilinx White paper WP 374, July 23, 2010.

[6] *Xilinx Partial Reconfiguration Guide ug702,* Xilinx Inc., v. 12.3, p. 103, 2010 [Online]. Available: http://www.xilinx.com.

[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] R. Dubey, *Introduction to Embedded System design Using Field Programmable Gate Arrays*, 1st ed. London: Springer-Verlag, 2009. ISBN 978-1-84882-015-9.

[9] P. Macgrefor and J. Squire, "Application of Microarray to the Analysis of Gene Expression in Cancer," *Clinical Chemistry*, vol. 48, no. 8, pp. 1170–1177, Aug. 2002.

[10] D. Brennan et al., "Application of DNA microarray technology in determining breast cancer prognosis and therapeutic response," *Expret Opin. Biol. Ther.*, vol.5, no. 8, pp. 1069–1083, Aug. 2005.

[11] 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. ISBN 1441912266.

[12] S. Cho and H. Won, "Machine Learning in DNA Microarray Analysis for Cancer Classification," *in Proc. of the First-Asia-Pacific Conf. on Bioinformatics*, Seoul, Korea, vol. 19, Feb. 4-7, 2003, pp. 189-198.

[13] V. Vapnik, *The Nature of Statistical Learning Theory*, 2nd ed., New York, US: Springer-Verlag, 2000. ISBN 0-387-98780-0.

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

[15] M. Murugappan, "Human emotion classification using wavelet transform and KNN," *in Proc. Int. Conf. of Pattern Analysis and Intelligent Robotics (ICPAIR)*, vol.1, no., pp.148-153, 28-29 June 2011.