AN FPGA BASED COPROCESSOR FOR CANCER CLASSIFICATION USING NEAREST NEIGHBOUR CLASSIFIER

Muhammad Atif Tahir

School of Computer Science
University of the West of England
Bristol, BS16 1QY, United Kingdom
muhammad.tahir@uwe.ac.uk

Ahmed Bouridane

School of Computer Science
Queen's University Belfast
BT7 1NN, United Kingdom
a.bouridane@qub.ac.uk

ABSTRACT

This paper discusses the suitability of reconfigurable computing to speedup classification problems using Nearest Neighbour (1NN) classifier. 1NN classifier is widely used in the literature especially in real-time applications such as face recognition, on-line hand-written character recognition and medical applications where the performance enhancement in terms of speed is desirable. To evaluate the effectiveness of our implementation on Field Programmable Gate Arrays (FPGAs), experiments were carried out on two medical data sets. Results have shown that the classification accuracy is exactly same for both FPGAs and microprocessor (μP) based solutions with FPGA has superior speed performances.

1. INTRODUCTION

The nearest-neighbour classifier (1NN) [1] has long been used in pattern recognition, exploratory data analysis, and data mining problems and has been proved to have good classification performance on a wide range of real-world data sets [2, 3]. 1NN classifier is also widely used in the literature for real-time applications such as face recognition [4, 5] and medical applications [6] where the performance enhancement in terms of speed is desirable. For all these application, a high-speed custom hardware is needed.

Although 1NN classifier is simple and provides a reasonable classification performance in most applications, the major drawback is that it is computationally intensive [7]. In order to find the nearest neighbour, a distance metric between the test sample and large number of samples in the training set is computed, thus making this approach quite computationally intensive.

The aim of this paper is to propose and implement efficient parallel architectures for 1NN classifier on Field Programmable Gate Arrays (FPGAs). FPGAs were originally developed for hardware circuit designs, however, FPGAs have become so dense and fast that they can be used as powerful reconfigurable computing systems for image processing algorithms [8, 9, 10, 11, 12]. Furthermore in this paper, the distance metric for 1NN classifier is mathematically simplified to save hardware resources and clock cycles.

The paper is organised as follows. Section 2 reviews existing hardware solutions for the nearest neighbour classifier. Section 3 mathematically reviews and simplified distance metric for 1NN classifier. Section 4 describes the proposed hardware architectures for 1NN classifier followed by experiments and discussion in Section 5. Section 6 concludes the paper.

2. PREVIOUS HARDWARE APPROACHES

Several hardware accelerators for the computation of the nearest neighbour rule have been proposed. These special-purpose systems are based on different approaches and have been optimised for specific classification problems. Zhou et al [13] describes a generic and fast classifier that uses a binary Correlation Matrix Memory (CMM) neural network for storing and matching a large amount of patterns efficiently, and a *K*NN rule for the classification. The CMM classifier has been tested on several benchmarks and when compared with a simple *K*NN rule, it gave less than 1% lower accuracy with a speed-up of over 4 times.

A new parallel NN classifier based on a 2-D Cellular Automation (CA) architecture was presented by Tzionas et al [14]. This approach is well suited for 2-D feature space and uses a network of processors to store the feature space. The main drawbacks are the large silicon area and the limited set of applications [7].

Lipman et al [15] proposed a parallel architecture for *K*NN classifier. Training samples are stored in memory and then the distances are computed and sorted in parallel. 64 distance calculation processors were used in a single chip. Multiple chips were required for more than 64 training samples. The main drawback is the use of a simple distance metric and the utilisation of multiple chips to store the training samples. A complete VLSI system for *K*NN classification was proposed by Ferrari et al [7]. The system adopts the exhaustive search by processing all the samples in the training database. Two different classification problems for handwritten recognition were chosen as benchmarks. The relative error is less than 1% using 16-bit fixed point arithmetic when compared with floating point calculation for handwritten-digit.

All the architectures proposed in the literature are either not providing accurate accuracy (For example, 1% error is unacceptable in medical data sets when compared with software approaches) or using a simple distance metric with large silicon area. A high-speed reconfigurable hardware is needed that provides accurate accuracy when compared with software approaches along with low cost solution. The aim of this paper is to use FPGA technology to speed-up the classification using 1NN classifier without any relative error.

3. DISTANCE METRIC FOR NEAREST NEIGHBOUR CLASSIFIER

In this work, neighbours are calculated using a squared Euclidean distance defined as: $D(x,y) = \sum_{i=1}^{f}(x_i - y_i)^2$ where x and y are two input vectors and f is the number of features.

Let x is the input vector for the test sample with f features and $y_1, y_2, y_3, ..., y_n$ are the input vectors of training samples of size n. The squared Euclidean distance of a test sample with a training sample vector y_1 is calculated as follows:

$$\begin{aligned} D(x, y_1) &= (x_1 - y_{11})^2 + (x_2 - y_{12})^2 + .. + (x_f - y_{1f})^2 \\ &= x_1^2 - 2x_1y_{11} + y_{11}^2 + x_2^2 - 2x_2y_{12} + y_{12}^2 + \\ &\quad .. + x_f^2 - 2x_fy_{1f} + y_{1f}^2 \\ &= x_1^2 + x_2^2 + .. + x_f^2 + x_1Y_{11} + x_2Y_{12} \\ &\quad + .. + x_fY_{1f} + K_1 \end{aligned} \quad (1)$$

where, $Y_{11} = -2y_{11}$, $Y_{12} = -2y_{12}$, $Y_{1f} = -2y_{1f}$ and $K_1 = y_{11}^2 + y_{12}^2 + .. + y_{1f}^2$. It is worth noting that y_{11}, y_{12}, .., y_{1f} are constants.

Similarly;

$$\begin{aligned} D(x, y_2) &= x_1^2 + x_2^2 + ... + x_f^2 + x_1Y_{21} + x_2Y_{22} + \\ &\quad ... + x_fY_{2f} + K_2 \end{aligned} \quad (2)$$

$$\begin{aligned} &... \\ &... \end{aligned}$$

$$\begin{aligned} D(x, y_n) &= x_1^2 + x_2^2 + ... + x_f^2 + x_1Y_{n1} + x_2Y_{n2} + \\ &\quad ... + x_fY_{nf} + K_n \end{aligned} \quad (3)$$

Finally, a test sample is assigned to the class with $min(D(x, y_1), D(x, y_2), ... , D(x, y_n))$. In equations 1-3, the term $(x_1^2 + x_2^2 + ... + x_f^2)$ is a common expression and by ignoring this expression, there is no effect on the final result as the minimum term in set $\{D(x, y_1), D(x, y_2), ... , D(x, y_n)\}$ will remain the same. Thus, the distance is calculated as follows:

$$D(x, y_1) = x_1Y_{11} + x_2Y_{12} + ... + x_fY_{1f} + K_1 \quad (4)$$

$$D(x, y_2) = x_1Y_{21} + x_2Y_{22} + ... + x_fY_{2f} + K_2 \quad (5)$$

$$\begin{aligned} &... \\ &... \end{aligned}$$

$$D(x, y_n) = x_1Y_{n1} + x_2Y_{n2} + ... + x_fY_{nf} + K_n \quad (6)$$

The simplification of equations 1-3 to equations 4-6 results in an efficient hardware by avoiding many substraction operations and using only constant multipliers instead of variable multipliers. For example, in equation 1, f multiplications, f additions, and f substractions are required, while in equation 4, only f constant multiplications, and $f + 1$ additions are required. Overall, with 1 training sample, $f - 1$ substraction are saved while for n training samples, $n * (f - 1)$ substraction are avoided, thus simultaneously using less hardware and saving clock cycles.

4. HARDWARE ARCHITECTURE FOR 1NN

The major steps for finding the class of unknown sample using 1NN are shown in Figure 1. At its most basic level, the programming model for classifying unknown sample is a host processor (typically a PC running at 2.4GHz Pentium 4-based system, programmed in C++). The host machine is working as a control unit and is responsible to load features vector in the external memory (SRAMs) of the FPGA.

The 1NN algorithm has two major computational phases: the distance metric calculation and the computation of the minimum distance. Distance calculations can be easily parallelised by assigning a processing element to each training sample. Figure 2 shows the

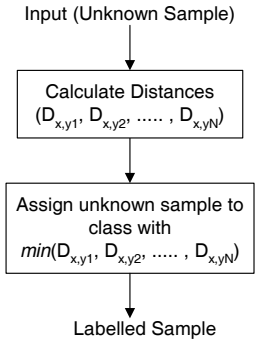

Fig. 1. Major steps during nearest neighbour classifier for finding the class of unknown sample.

hardware architecture for 1NN with feature vector of size F. The original feature vector (unknown sample) is loaded into the external memory (SRAM-BANK0) of the FPGA board. The training samples are stored in F parallel Block ROMs. At each iteration, the features of an unknown sample are multiplied by the corresponding features of F training samples. Thus, F Processing Elements (PEs) are executed in parallel to perform the multiplication and accumulation with the accumulators in PEs are initialized with constants $(K_1, K_2, ... , K_N)$. A set of comparators are then used to find the minimum distance and simultaneously, the features of unknown sample are multiplied by F new training samples of the next iteration. The process is iterated N/F times where N is the number of samples. Finally, the unknown sample is assigned to the class having the minimum distance.

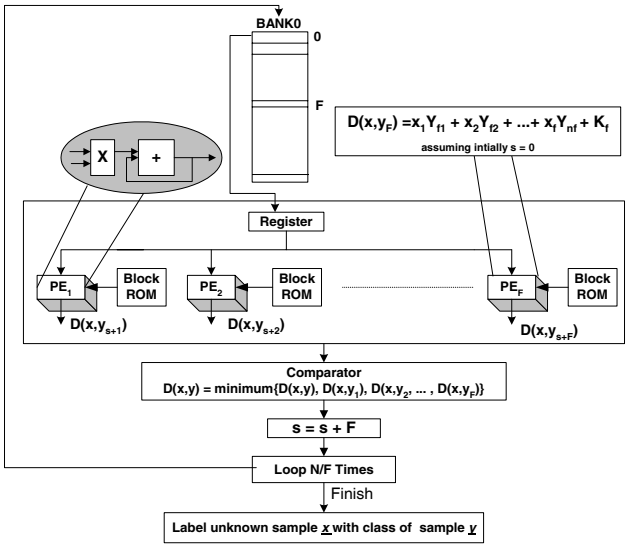

Fig. 2. Hardware Architecture for Nearest Neither Classifier. F = Number of Features. N = Number of Training Samples. Initially s = 0.

In addition to parallelism, pipelining is also used in the implementation to improve the performance. Figure 3 demonstrates an example how pipelining is used for calculating the distance metric. It can be clearly seen from the figure that before pipelining $4 * F$ clock cycles are required for distance calculation of a training sample with a delay of 1 clock cycle for the multiplication operation

and F features. This delay has been eliminated by using pipelining where the number of clock cycles has been reduced to $F + 3$.

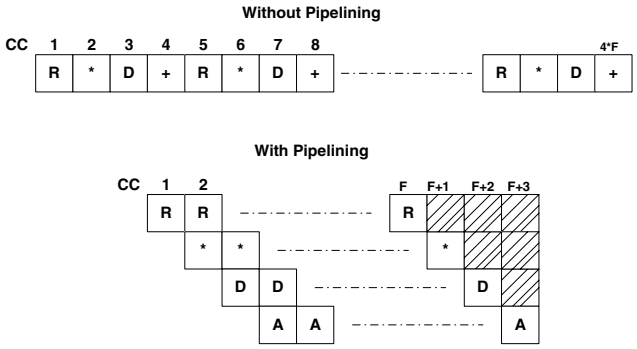

Fig. 3. An example showing pipelining for distance calculation of a training sample. R = Read, *=Multiplication, D=Delay, +=Add, CC=Clock Cycles.

The proposed architecture has been implemented by using Handel-C [16] and Xilinx Coregen [17]. Handel-C is a truly innovative C-like language for implementing algorithms in hardware. The output from Handel-C is a file that is used to create the configuration data for the FPGA. The target hardware for this work is Celoxica RC1000-PP PCI based FPGA development board equipped with a Xilinx XCV2000E Virtex FPGA having 19,200 slices and 655,360 bits of block RAM, and four banks of static RAM with 2MB each [16, 17].

5. EXPERIMENTS AND DISCUSSION

To evaluate the effectiveness of our implementation on FPGAs, experiments were carried out on two medical data sets reported in [18, 6]. A short description is mentioned below:

- **Breast Cancer data set:** This data set Wisconsin Diagnostic Breast Cancer (WDBC) [18] consists of 569 samples and divided into two 2 groups: 357 benign and 212 malignant. The number of features is 30.
- **Prostate Cancer data set:** This data set is derived from prostatic nuclei extracted from prostate tissue [6]. This data set consists of 230 samples and labeled into 3 classes: 63 cases of Benign Prostatic Hyperplasia (BPH), 79 cases of Prostatic Intraepithelial Neoplasia (PIN) and 88 cases of Prostatic Carcinoma (PCa). The number of features is 13.

Due to the different range of values for the original features, the input feature values are normalised over the range [1,10] using Equation 7 [19]. Normalising the data is important to ensure that the distance measure allocates equal weight to each variable. Without normalization, the variable with the largest scale will dominate the measure. Leave-one-out method has been used for cross-validation.

$$x_{i,j}^{'} = \left(\frac{x_{i,j} - min_{k=1\ldots n} x_{(k,j)}}{max_{k=1\ldots n} x_{(k,j)} - min_{k=1\ldots n} x_{(k,j)}} * 9\right) + 1 \qquad (7)$$

where $x_{i,j}$ is the j^{th} feature of the i^{th} pattern, $x_{i,j}^{'}$ is the corresponding normalized feature, and n is the total number of patterns.

Tables 1 and 2 show the overall classification error for the breast and prostate cancer data sets respectively. The classification error is exactly the same for both μ-P and FPGA based solutions when using 14-bit and 10-bit numbers. For the breast cancer data set, there is a difference in classification accuracy when 6-bit fixed point numbers are used. For prostate cancer data set, there is difference in classification accuracy when 2-bit fixed point numbers are used. This is mainly due to some loss of arithmetic precision during the implementation. Although, the classification accuracy is improved using 6-bit fixed point numbers in breast cancer data set, our main aim is the comparison of the classification accuracy between hardware and software approaches. From the results, it can be concluded that, for up to 10-bit fixed point numbers, a hardware implementation is suitable for these type of data sets.

Table 1. Comparison of Classification Error using μ-P and FPGA for breast cancer. CL = Classified as, B=Benign, M=Malignant, E=Error, O = Overall. FP = Floating Point, FxP = Fixed Point. F=30.

CL	μ-P/FPGA (32-bit FP/14/10)			FPGA (6-bit FxP)		
	B	M	E(%)	B	M	E(%)
B	342	15	4.09	344	13	3.78
M	12	200	5.66	12	200	5.66
O			4.88			4.72

Table 2. Comparison of Classification Error using μ-P and FPGA for prostate cancer. CL = Classified as, B= BPH, P= PIN, C=Cancer, E=Error, O = Overall. FP = Floating Point, FxP = Fixed Point. F=13.

CL	μ-P/FPGA (32-bit FP, 14/10/6-bit FxP)				FPGA (2-bit FxP)			
	B	P	C	E(%)	B	P	C	E(%)
B	62	0	1	1.59	61	0	2	3.17
P	0	79	0	0	1	78	0	1.26
C	1	0	87	1.14	5	2	81	7.95
O				0.91				4.13

Table 3 shows the execution time comparison between an μP-based and an FPGA-based implementation of 1NN classifier for breast and prostate cancer data sets using one test sample. The result shows that the performance of an FPGA implementation is approximately 15, 38 and 47 times faster than Pentium 4 PC using Virtex-E, Virtex-2, and Virtex-2P respectively for breast cancer data set. For the prostate cancer, the performance of an FPGA implementation is approximately 6, 17, and 19 times faster than μ-P using Virtex E, Virtex-2, and Virtex-2P, respectively. This improvement in the performance depends upon 2 factors: feature vector size (F) and the number of training samples (N). For the breast cancer problem, $F = 30$, and $N = 569$, thus, 30 parallel operations (multiplications and additions) are performed at each iteration with the maximum number of iterations of 19. Because of the larger values of F and N, the performance of FPGA is best in the breast cancer data set when compared with prostate cancer data set. Tables 4-5 show the clock speed and the area used for different fixed point implementations on FPGAs using various device families. Note that the implementations using Virtex-2 and Virtex-2P result in the same area in terms of the number of CLBs.

5.1. Block RAM vs Number of Training Samples

The major problem in this proposed architecture is the size of Block RAM. The number of training samples depends upon the size of

Table 3. Execution Time in μsec between μ-P and FPGA implementation of 1NN classifier using Xilinx Device Families. BC = Breast Cancer, PC = Prostate Cancer, DF = Device Family, FP = Floating Point, FxP = Fixed Point.

Data set	μ-P	DF	FPGA (14-bit FxP)	FPGA (10-bit FxP)	Speed-Up
BC	180.1	Virtex-E	13.5	11.8	13.34 - 15.26
		Virtex-2	7.50	4.71	24.0 - 38.21
		Virtex-2P	6.27	3.83	28.72 - 47.01
PC	32.5	Virtex-E	6.13	5.23	5.30 - 6.21
		Virtex-2	3.41	2.08	9.52 - 15.6
		Virtex-2P	2.85	1.75	11.4 - 18.6

block RAM within the FPGA device. One possible solution is the use of dedicated RAMs to store the training samples as proposed by [15, 7]. However, new advancement in FPGA technology has lead to an increase of the size of BLOCK RAM. For example, the Xilinx Virtex-4 FPGAs supports up to $9,936Kb$ of Blocks RAM as compared to only $640Kb$ of Block RAM in Virtex-E devices used in this paper. It is estimated that the new Virtex-4 devices can store up to 8000 training samples with each training sample consisting of feature vector of size 16. Thus, FPGAs can be used as a co-processor in many medical data sets as the number of training samples are less than 8000 in most medical data sets.

Table 4. Clock Speed and area used for classification using 1NN for breast cancer data set using various Xilinx Device Families. $F = 30$.

Device Family		FPGA (14-bit)	FPGA (10-bit)
Virtex-E	Clock Speed (MHz)	50.0	57.2
	Number of occupied Slices	10,967	8,061
	Total number of 4 input LUTs	19,397	14,005
	Number of block rams	150	120
Virtex-2 Virtex-2P	Clock Speed (MHz)	90.0	143.2
		107.7	176.2
Virtex-2 & 2P	Number of occupied Slices	4,454	3,141
	Total number of 4 input LUTs	6,617	4,405
	Number of block rams	90	60
	Number of $18 * 18$ multipliers	120	30

6. CONCLUSION

Reconfigurable architectures have many applications in classification algorithms. Depending on the classifier, reconfigurability can assist in speeding up classification process. In this paper, FPGA is used as a co-processor to accelerate the classification using 1NN classifier. To evaluate the effectiveness of our implementation on FPGAs, experiments were carried out on two data sets. Results have shown that the classification accuracy is exactly same for both FPGAs and μP based solutions with FPGA has superior speed performances (average up to 33 times faster).

7. REFERENCES

[1] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[2] C. Domeniconi, J. Peng, and D. Gunopulos, "Locally adaptive metric nearest-neighbor classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1281–1285, 2002.

Table 5. Clock Speed and area used for classification using 1NN for prostate cancer data set using various Xilinx Device Families. $F = 13$.

Device Family		FPGA (14-bit)	FPGA (10-bit)
Virtex-E	Clock Speed (MHz)	50.2	58.9
	Number of occupied Slices	4,311	3,079
	Total number of 4 input LUTs	7,971	5,635
	Number of block rams	39	26
Virtex-2 Virtex-2P	Clock Speed (MHz)	90.2	147.8
		108.1	180.3
Virtex-2 & 2P	Number of occupied Slices	1,751	1,206
	Total number of 4 input LUTs	2,888	1,930
	Number of block rams	26	26
	Number of $18 * 18$ multipliers	52	13

[3] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, *Machine Learning, Neural and Statistical Classification*, Ellis Horwood, 1994.

[4] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," *ACM Computing Surveys*, vol. 35, no. 4, pp. 399–458, December 2003.

[5] R. Chellappa, C. L. Wilson, and S. Sirohey, "Human and machine recognition of faces: A survey," *Proceedings of the IEEE*, vol. 83, no. 5, pp. 705–740, May 1995.

[6] M. A. Tahir, A. Bouridane, F. Kurugollu, and A .Amira, "A novel prostate cancer classification using intermediate memory tabu search," *EURASIP Journal on Applied Signal Processing*, vol. 14, pp. 2241–2249, 2005.

[7] A. Ferrari, M. Borgatti, and R. Guerrieri, "A complete system for NN classification based on a VLSI array processor," *Pattern Recognition*, vol. 33, pp. 2083–2093, 2000.

[8] A. Bouridane, D. Crookes, P. Donachy, K. Alotaibi, and K. Benkrid, "A high level fpga-based abstract machine for image processing," *Journal of Systems Architecture*, vol. 45, no. 10, pp. 809–824, 1999.

[9] D. Crookes et al., "Design and implementation of a high level programming environment for FPGA-based image processing," *IEE Proceedings on Vision, Image and Signal Processing*, August 2000.

[10] F. Bensaali, A. Amira, and A. Bouridane, "Accelerating matrix product on reconfigurable hardware for image processing applications," *IEE Proceedings on Circuits, Devices and Systems*, vol. 152, no. 3, pp. 236–246, June 2005.

[11] M. A. Tahir, A. Bouridane, and F. Kurugollu, "An FPGA based coprocessor for GLCM and Haralick texture features and their application in prostate cancer classification," *Analog Integrated Circuit and Signal Processing*, vol. 43, pp. 205–215, 2005.

[12] M. A. Tahir, A. Bouridane, and F. Kurugollu, "An FPGA based coprocessor for the classification of tissue patterns in prostatic cancer," in *14th International Conference on Field Programmable Logic and its Applications (FPL), Lecture Notes in Computer Science*. 2004, pp. 771–780, Springer Verlag.

[13] P. Zhou, J. Austin, and J. Kennedy, "A binary correlation matrix memory k-NN classifier with hardware implementation," in *Proceedings of the Ninth British Machine Vision Conference*, 1998.

[14] P. G. Tzionas, P. G. Tsalides, and A. Thanailakis, "A new, cellular automaton-based, nearest neighbor pattern classifier and its VLSI implementation," *IEEE Transactions on VLSI*, vol. 2, no. 3, 1994.

[15] A. Lipman and W. Yang, "VLSI hardware for example-based learning," *IEEE Transactions on VLSI*, vol. 5, no. 3, 1997.

[16] URL, "http://www.celoxica.com," .

[17] URL, "http://www.xilinx.com," .

[18] C. Blake, E. Keogh, and C. J. Merz, "UCI Repository of machine learning databases," University of California, Irvine.

[19] M. L. Raymer et al., "Dimensionality reduction using genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 2, pp. 164–171, 2000.