An Accelerated VLSI Approach for K-Means Clustering Algorithm for the application of Microarray Data Analysis

Ila Roy Saxena, Vikas Pathak, Manju Choudhary, Himanshu Sharma
Department of Electronics and Communication, Swami Keshvanand Institute of Technology, Management & Gramothan Jaipur-302017, (INDIA)
Email- illaroy611@gmail.com

Received 23.08.2019 received in revised form 16.12.2019, accepted 23.12.2019

Abstract : This paper proposed a parallel and pipelined architecture for faster execution of k-means clustering algorithm for microarray data analysis. This is verified by calculating the computational time of algorithm when run on XILINX ISE 14.4 and MATLAB. The computational time taken by MATLAB simulation it is 14.2390ms while for VLSI simulation is found to be 0.0754ms hence VLSI simulation is 188 times faster than MATLAB Simulation. Also, computational time of proposed work which is 0.0754 ms, is compared with the previous work time which is 0.5891 ms and it is found that proposed work is 7 times faster than previous work.

Keywords : Clustering, k-means algorithm, microarray chip, bioinformatics.

1. INTRODUCTION

The data mining techniques have been quite interesting research area that lured scientists and researchers to explore genetic world from a close insight. It basically involves method at intersection of machine learning, statistics and database systems. Data mining is also known as knowledge discovery in database or knowledge mining. All of such techniques aim to find some similarity or pattern hidden in large datasets [1]. Raw data is processed and converted into a more digestible form known as information. Further this processed form i.e., information can be used to make decisions or to draw a meaningful conclusion. Most of the data mining techniques include clustering, classification, neural network, association etc. In this paper, clustering is being focused. Clustering is an unsupervised learning method in which no set of rules are applied before grouping data into different clusters. Clustering basically divided the data into groups that show some similarity within it but are dissimilar from other groups [2].

K-Means is one such algorithm used for the purpose of clustering [3]. Earlier it was applied mainly to the field of image analysis to segment the image, to find the region of interests in medical imaging to indentify the tumor cell and its subtype's etc. New application includes document clustering, fraud detection, genomics study etc. Microarray techniques are used to understand/interpret data generated from experiments on DNA/RNA/proteins. It is also known as Gene chip analysis [4]. The advantages offered are that it is helpful in understanding the functions performed by genes in an organism, investigating the expression of large number of genes in a single experiment. It empowered scientist to learn about fundamental processes or cycles involved at various development and growth stages of a cell. Also, by exploring things at genetic level may helps to comprehend the genetic causes of anomalies occurring in an organism especially human. This will not only allow early diagnostic but also better treatment [5].

This paper focuses on use of K-mean clustering algorithm to cluster yeast microarray data in a more efficient manner. The need for improving performance while executing algorithm lies in the fact that the microarray data is growing exponentially. In the proposed work efficiency is increased in terms of speed (computational time taken by the algorithm) by using concepts of parallelism and pipelining. This paper proposed a parallel and pipelined architecture and based on that VHDL code for k-means clustering algorithm has been written and synthesized using Xilinx ISE 14.4.

Section-2 gives the basic introduction of background of clustering, fixed point representation and microarray techniques. Section-3 describes the methodology used to achieve this objective. In section-4 various simulations and synthesis results of VLSI and MATLAB simulation of the code written for the algorithm. Finally, section-5 concludes the paper.

2. BACKGROUND

2.1 Clustering

Clustering is an unsupervised method of machine learning. By unsupervised it is meant that clustering is analogous to learning without teacher or analyzing things without any predefined set of rules i.e., finding the inherent patterns or trends hidden inside the raw or unprocessed data. There are various ways to cluster data as given below

1. Partitioning methods
2. Hierarchical methods
3. Density based methods
4. Model based methods
5. Grid based methods

K-means algorithm falls in the category of partitioning methods. It is an iterative algorithm which divides a set of data into k number of clusters. The basic functional steps of the algorithm are explained below with the help of a flow chart.

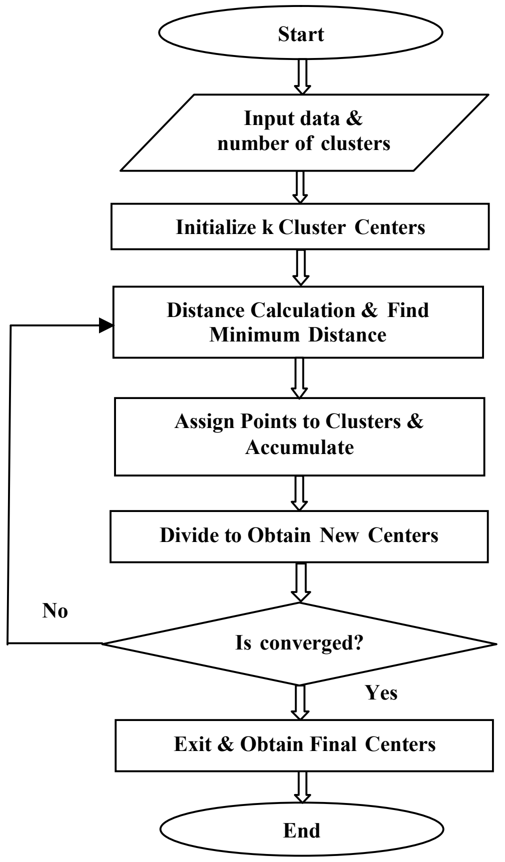
Start
Input data & number of clusters
Initialize k Cluster Centers
Distance Calculation & Find Minimum Distance
Assign Points to Clusters & Accumulate
Divide to Obtain New Centers
No
Is converged?
Yes
Exit & Obtain Final Centers
End

Figure 1 : Flow Chart of K-means Clustering Algorithm

Step1: Initialize the cluster centers.
Initialize the k number of centers randomly the given data set.
Step2: Distance calculation & find minimum distance and cluster index.

Manhattan distance formula has been used in this dissertation to calculate distance between data points and cluster centers. It is derived from Minkowski distance formula (a generalized distance formula) given below by equation 1. Also, other advantages offered by Manhattan distance over Euclidean distance (when r=2) is that it reduces the complexity in terms of square multiplication which requires more logical resources. In this step, minimum distance for each data point is also found by comparing the computed distance. Based on the minimum distance found, a cluster index value is also updated. Cluster index value indicted the cluster number for which minimum distance has been found.

$$\textbf{Distance}(\mathbf{d_k}) = \left(\sum_i^n |p_k - q_k|^r\right)^{1/r} \ldots\ldots\ldots\ldots\ldots\textbf{(1)}$$

Where, r is a parameter whose value determines type of distance for e.g., r=1 gives Manhattan distance and r=2 gives Euclidean distance

n is total number of attributes

p_k , q_k are the kth vector attributes of the two vectors being compared.

Step3: Accumulation of data points

Now assign the data points to their respective cluster and accumulate (add) them according to cluster index value. Also, a counter value is also used which keeps the account for number for data points assigned to each cluster.

Step4: Division to obtain new centers

In the next step, accumulated points are divided by their respective counter values for each cluster so as to obtain new centers.

Step5: Repeat step2 to step4 until the cluster centers stop moving or after a specified number of iterations.

2.2 Fixed Point Arithmetic

There are two methods to represent a real number for the use of computing namely, fixed point and floating point. In the fixed-point notation [6] there are fixed number of digits after the decimal point whereas in floating point there are varying numbers of digits after the decimal point. A scaling factor is used in fixed point representation to represent a fractional number. It is basically represented as an integer scaled by some factor which is mostly some power of 10 or power of 2. For example, 7.51 can be represented as 7510 with a scaling factor of 1/1000 in fixed point format.

In order to make use of fixed-point arithmetic in VHDL there are packages and library available which are needed to be included in the main library. Another notation called Q-Format is also utilized to represent fractional value. For instance, if fractional number consists of three bits of integer part and four bits of fractional part then this number will be represented as Q3.4. For negative numbers it is necessary first to take their two's complement and then represent them in fixed point format. Fixed point makes use of implied binary point just like decimal point which is not specified in the hardware, programmer needs to take care of it to comprehend the result of calculations accurately [7]. Fixed point format is used when it is required that the designed system works fast enough and consumes less memory and hence efficiency is increased.

2.3 Microarray Techniques

A DNA microarray is a collection of single stranded DNA (ssDNA) probes immobilized on a coated silicon glass in an organized grid fashion. Each spot on the chip consists of a known gene sequence known as probes. It is basically a two-dimensional array that consist of several thousands of specified known gene sequences on it which are used to detect the expressed genes in a sample of complementary DNA (cDNA) of some organism collected under certain conditions. This detection is useful for gene expression profiling. Gene expression profiling helps to understand the cellular function by measuring levels of expressed messenger RNA (mRNA) in a sample. It can also be useful in understanding how the cells reacted to a particular treatment.

3. PRE-PROCESSING AND METHODOLOGY ADOPTED

3.1 Pre-processing

3.1.1 Fixed point range analysis

The equation (2) given below calculates number of bits required to represent the integer part (QI) of binary fixed-point number

$$QI \geq [\log_2(| \text{range} |)] \qquad \text{….(2)}$$

Range is basically the differences between maximum and minimum value in the data set. In the microarray yeast data used in the algorithm, maximum value is 4.216 and minimum value is -6.4032. So range will be 10.619 and from equation (2), it has been found that six bits will be required to represent the data points in the yeast dataset.

3.1.2 Precision Analysis

The equation (3) given below calculates number of bits required to represent the fractional part (QF) of binary fixed-point number

$$QF \geq [\log_2(1/ \text{precision})] \qquad \text{….(3)}$$

In the proposed method, precision of 0.001 is required so from equation (3), it has been found that ten bits are required for the fractional part of data points.

The wordlength required to represent each data point is given by QI+QF, therefore sixteen bits are required to represent this input data. The filtered yeast data matrix is of size 187 X 7 i.e., there are 187 rows representing log2 ratios of mRNA levels in the yeast measured at seven different time (in hrs) instances represented in the seven columns. Therefore, there is requirement of seven FIFO to store this data with size of each FIFO being 256 X 16 bits.

3.3 Methodology

The methodology adopted is presented in the below given fig.2 and also discussed step by step.

1. The yeast value data set is loaded in MATLAB. It is also available freely on online website of NCBI [8]. This data is from microarray study of gene expression in yeast, published by DeRisi et. all. 1997 [9]. The DNA microarray was used to study temporal gene expression of genes present in yeast during diauxic shift which includes shift from fermentation to respiration in yeast metabolism.
2. The noise present in the data in the form of empty cells, missing profiles, zero expression profiles, low variance value are removed using MATLAB functions like genevarfilter, genelowvalfilter etc. Due to this filtering dataset from 6400 X 7 matrix is reduced to 187 X 7 matrix.
3. In the next step this floating-point data is converted into fixed point data.
4. First, simulation of MATLAB code for K-Means algorithm (which is sequential in nature) is done. This is required to show the speed achieved when algorithm is simulated in Xilinx. This will also help to understand the basic flow of algorithm.
5. For VLSI simulation, it is first required that input data to be written in FIFO. This could be accomplished by writing data into text file format with the help of 'csvwrite' function in MATLAB. A data point in yeast data consists of seven distinct features. Therefore, there are seven text files consisting of list of distinct features for each data point.
6. Next a test bench is written to read and write FIFO.
7. As soon as the FIFO is written and read, other modules such as distance finder block, minimum distance finder block, accumulator,

divider etc. starts functioning.

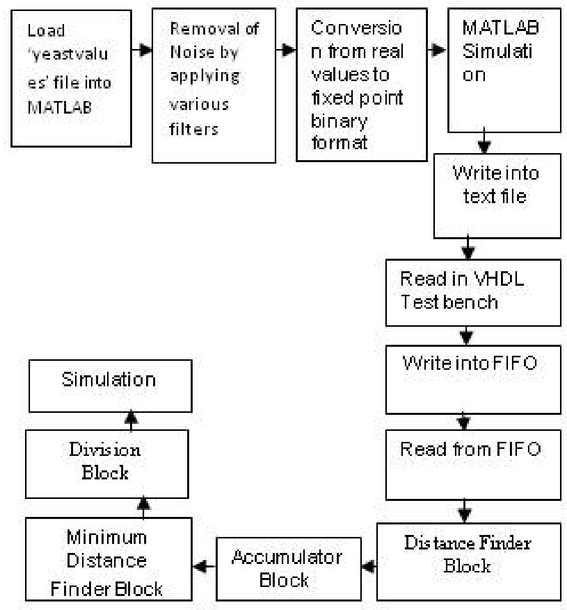
Load 'yeastvalues' file into MATLAB
Removal of Noise by applying various filters
Conversion from real values to fixed point binary format
MATLAB Simulation
Write into text file
Read in VHDL Test bench
Write into FIFO
Read from FIFO
Distance Finder Block
Accumulator Block
Minimum Distance Finder Block
Division Block
Simulation

Figure 2: Methodology to execute K-Means algorithm

4. PROPOSED ARCHITECTURE

This dissertation proposes a parallel and pipelined architecture/module for K-means clustering algorithm to improve its performance in terms of speed as shown in the fig.3. This architecture consists of four blocks FIFO block, Minimum Distance Finder, Accumulator and Divider.

FIFO is used to store the input data instead of BRAM (Block RAM) as done in the previous work. In BRAM, data is first written than reading/fetching of data starts. On the other hand, in the proposed work, FIFO block starts reading data after one clock cycle of data write thus saving a lot of time and enabling parallelism. Since the data consists of seven features hence seven FIFOs has been used. Simultaneously distance between data points and centers are computed in the second block with latency of one clock cycle.

The distance found is fed to comparator and cluster index based on minimum distance found is obtained at the same. Also, the index value fed to demux enables corresponding accumulator to start accumulating data points. The demux also enables the corresponding counter which gives the count of data point belonging to clusters with a latency of one clock cycle.

The distance finder block calculated all distance between a data point and six cluster centers in one clock cycle. Similarly, accumulator block starts accumulating point as soon as demux enables it. This shows that parallelism significantly doubled the throughput from previous work [44].

When final values of accumulation and counter are obtained, they are fed to divider block as input so as to obtain new centers. Instead of using Divider IP core as done in the previous work, this dissertation used divider as a component. IP (intellectual property) core are preconfigured logic functions giving less flexibility to user to parallelize the code while using divider as a component gives more flexibility. This divider was active for k number of cycles calculating new centers sequentially but in the proposed work, divider is active for just one clock cycle i.e., calculating the centers simultaneously.

5. RESULTS

5.1 MATLAB Results

The MATALB simulation was done on an Intel core i5, 2.30 GHz CPU with 4GB RAM running on Windows 10 operating system. The 'tic-toc' command has been used for more than 100 runs of the algorithm to obtain computational time taken in MATLAB simulation of the algorithm.

The average execution time recorded was 14.2390 ms with minimum execution time of 12.3530 ms and maximum execution time of 16.1250 ms. For this result, initial centers are pre defined and given as an input to the algorithm.

5.2 VLSI Results

The synthesis results for VHDL code of k-means clustering algorithm are generated on Xilinx for Virtex-7 XC7VX1140T-2FLG1930 device. The synthesis results include device utilization summary, logic utilization summary and timing summary as shown below in table 1, 2 and 3 respectively.

Table1: Device Utilization Summary

S. No	Parameters	Available	Utiliz-ed	% of Utilization
1.	No. of slice register	1424000	5315	0%
2.	No. of LUTs	712000	105197	14%
3.	No. of flip-flop pair	106466	4046	3%
4.	No. of I/O Blocks	1100	788	71%
5.	No. of Block RAM/FIFO	1880	4	0%
6.	No. of BUFG	128	7	5%

Below given table 2 shows different logic consumed by each of the individual blocks.

Table2: Logic Utilization Summary

S.No	Hardware Units	Number
1.	256x16 bit dual port RAM	7
2.	Adders/Subtractor	3952
3.	Registers	171
4.	Latches	2646
5.	Comparators	1863
6.	Multiplexers	73997
7.	FSMs	1
8.	XOR Gate	612

The next table 3 shows various timing parameters generated in the synthesis report.

In order to obtain the time taken to execute the K-Means clustering algorithm in Xilinx, two parameters are required first the simulation result obtained using Isim simulator, and other the clock frequency as depicted in timing summary of synthesis result, and then apply them in equation (4) as shown below.

Table 3 : Timing Summary from the Synthesis Report

S.No	Parameters	Parameters Value
1.	Minimum period/critical path delay	22.171ns
2.	Minimum input arrival time before clock	1.719ns
3.	Maximum output required time after clock	0.798ns
4.	Maximum combinational path delay	No path found
5.	Maximum clock frequency	45.104MHz

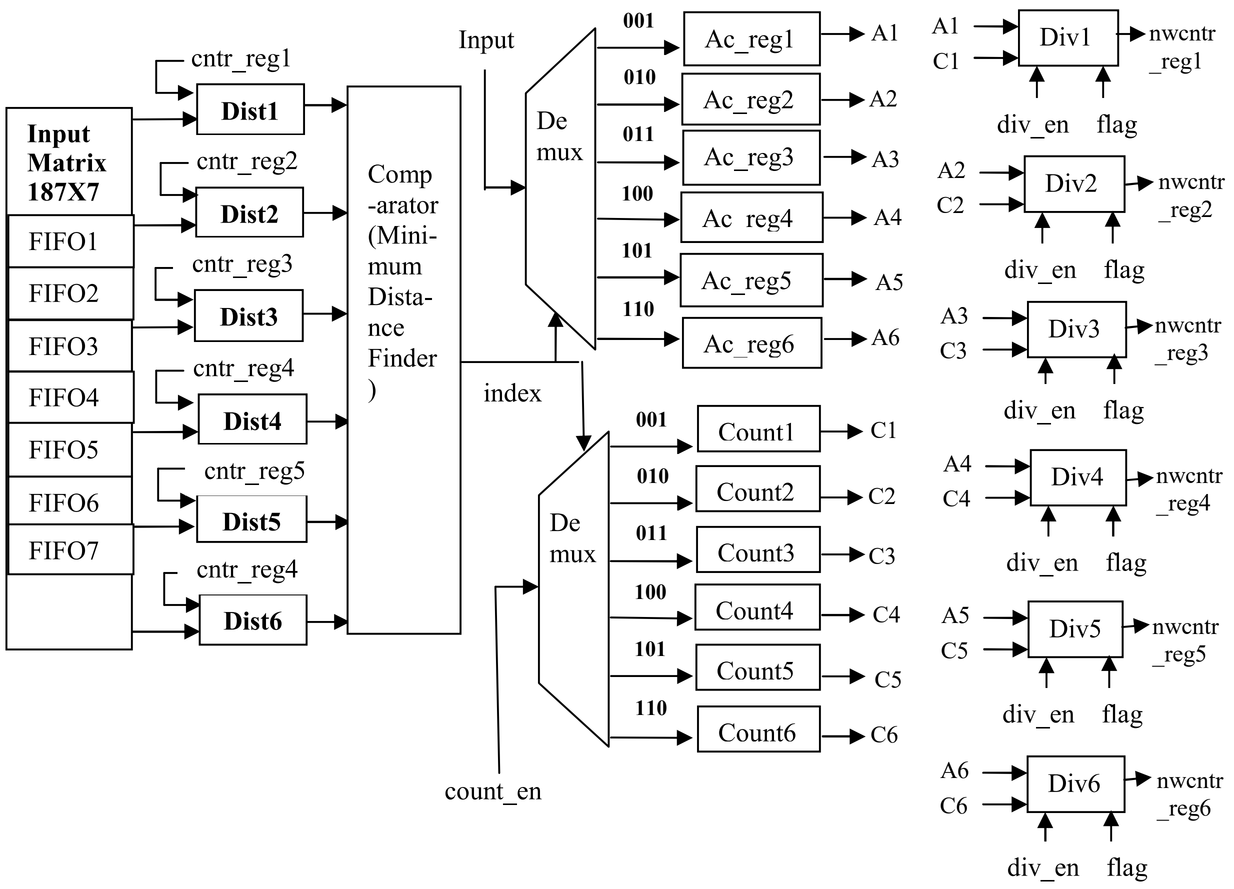

Figure 3: Proposed Architecture for k-means clustering algorithm

Execution time=

$$\frac{\text{Clock cycles per iteration * total iterations (in ms)}}{\text{Maximum clock frequency (MHz)}} \quad \textbf{(4)}$$

Total_iterations = 18

Clock cycles per iteration = 189

Maximum clock frequency = 45.104 MHz

Execution time = 0.0754 ms

5.3 Verification and Comparison of Computational Time of VLSI and MATLAB Simulation

The table 4 summarizes the speed up achieved during VLSI simulation of the algorithm in comparison to MATLAB simulation.

The MATLAB execution time is found to be 14.2390 ms while for VLSI simulation of the algorithm it is 0.0754 ms. Hence it has been observed that VLSI simulation is 188 times faster than MATLAB simulation.

Table 4 : Comparison of Computational time of VLSI and MATLAB simulation for 187x7 microarray datasets of yeast

Execution time (MATLAB)	Execution time (Xilinx)	Speed up
14.2390 ms	0.0754 ms	188

The table 5 as shown above compares the computational time of the proposed work with previous work [10]. The execution time obtained in the previous work is found to be 0.5891ms while for the proposed work it is 0.0754ms.

Hence it is observed that performance (in terms of computational time) of the algorithm has been improved in the proposed work and algorithm runs 7 times faster than previous work.

Table 5 : Comparison of Computational time of proposed work and previous work

Execution time (Previous work [10])	Execution time (Proposed work)	Speed up
0.5891 ms	0.0754 ms	7.81

6. CONCLUSION

This dissertation proposed a parallelized and pipelined architecture for K-means clustering algorithm to improve its performance in terms of speed for the application of microarray data analysis. Based on this architecture, VHDL code has been written and synthesized using Xilinx 14.4 and simulated using Isim Simulator. The number of clusters is chosen to be six. All four blocks have been pipelined to increases overall instruction throughput. MATLAB code which is sequential in nature has also been written to show the speed up achieved by comparing the computational time taken by the algorithm during MATLAB simulation with that of obtained through VLSI simulation. Also, computational time of proposed work which is 0.0754 ms, is compared with the previous work time which is 0.5891 ms and it is found that proposed work is 7 times faster than previous work.

REFERENCES

[1] N. Jain and V. Srivastav, “Data Mining Technique: A Survey Paper”, International Journal of Research in Engineering & Technology, vol. 2, pp 116-119, 2013.

[2] J. Han and M. Kanbu, “Data Mining Concepts & Techniques”, Cluster Ananlysis: Basic Concepts and Methods, Chapter-10, Morgan Kaufman Publication, 3rd edition, USA, 2012.

[3] Dr. R.M. Suresh, K. Dinakaran and P. Valarmathie, “Model Based Modified K-Means Clustering for Microarray Data”, International Conference on Information Management and Engineering Malaysia, pp. 271-273, 2009.

[4] B. SivaLakshmi and N. M Rao, “Microarray Image Analysis using K-Means Clustering Algorithm”, International Journal of Research in Advent Technology, vol. 6, no. 12, pp. 3796- 3802, 2018.

[5] P.F. Macgregor and J.A. Squire, “Application of Microarrays to the Analysis of Gene Expression in Cancer”, Clinical Chemistry, vol. 48, pp. 1170-1177, 2002

[6] E. L. Oberstar, “Fixed Point Representation and Fractional Math”, IEEE Transaction on Circuits and System, vol. 41, no. 11, pp. 1-18, 1994.

[7] S. Ramanathan, G. Vinothini, A. C. Chaudhari and Dr. K. Sivasankaran, “Design and Implementation of Fixed Point Arithmetic Unit”, International Journal of Engineering Research and Application, vol. 6, no. 6, pp.11-13, 2016.

[8] 14 hrs Microarray yeast datasets, [datasets online] available: http:/www.ncbi.nlm.nih.gov/querry/acc.cgi.

[9] J. L. DeRisi, Vishwanath R. Iyer and Patrick O. Brown, “Exploring the metabolic and genetic control of gene expression on a genomic scale”, Journal of American Association for the Advancement of Science, vol. 278, pp. 680-686, 1997.

[10] H. M. Hussain1, K. Benkrid1, H. Seker, and A. T. Erdogan, “FPGA Implementation of K-means Algorithm for Bioinformatics Application: An Accelerated Approach to Clustering Microarray Data”, Conference on Adaptive Hardware and Systems, pp. 248-255, 2011.