Digital System Design

we gather, we share, you learn..

www.digitalsystemdesign.in

A Novel FPGA Implementation of K-means Algorithm*

Shirshendu Roy

March 24, 2020

Abstract

K-means algorithm is a well known algorithm in the field of machine learning or big data analysis. This algorithm takes huge data set and segregate the similar data samples into K groups or clusters. In many application, real time implementation of K-means algorithm is very important. Thus Field Programmable Gate Array (FPGA) devices are suitable for implementing K-means clustering algorithm. In this work, a novel architecture of the K-means algorithm is proposed. A basic prototype of K-means algorithm is implemented on NEXYS DDR2 (xc7a100t) FPGA device. A detail analysis of the proposed design is also presented in this work.

Keywords: Field Programmable Gate Array (FPGA), Machine Learning, K-means Algorithm, Euclidean Distance

1 Introduction

Clustering is an important data analysis technique to know the structure of a big data. The objective of this technique is to segregate the similar data samples in some sub groups. The knowledge of homogeneous groups can lead to apply many other techniques to optimize the data set. There are many clustering algorithms exists in literature but in this work K-means [1] algorithm is discussed.

K-means algorithm is very popular because of its simplicity. This is an iterative algorithm which tries to partition a dataset into K groups. K-means algorithm tries to group the similar data samples in a cluster in every iteration. This is carried out based on some formulas which computes the similarity between two vectors. This algorithm results K homogeneous groups which are non-overlapping means no data samples can belong to two or more groups.

In many application, real time clustering of the data samples are required. In this context,K-means algorithm is needed to be implemented on some hardware to support real time clustering. In this work, a novel architecture of K-means algorithm is proposed and this architecture is implemented on Field Programmable Gate Array (FPGA) based platform. The major objective of this work to present an hardware efficient architecture of K-means algorithm without hampering the execution time.

The arrangement of the manuscript is as follows: Section 2 describes the K-means algorithm and illustrates the steps in details. In the section 3, the propose architectures is discussed in details. Here the sunblocks are discussed in details. In the section 4, the design performance is given with hardware and timing analysis. The conclusive remarks on the findings of the research are given in section 5.

Algorithm 1 K-means Algorithm

Input: Data Matrix $A \in \mathcal{R}^{M \times N}$, number of clusters ($K$) and number of iterations (I).
Output: K clusters with its centroids ($cntd$).

1: **Initialization** Randomly selects K number of elements from the matrix A. Initially set them as $cntd^k = A_k$ for $k = 1$ to K. Initially $cltr^k$ is a null vector for $k = 1$ to K.
2: **for** $i \leftarrow 1$ to I **do**
3: **for** $j \leftarrow 1$ to N **do**
4: **for** $k \leftarrow 1$ to K **do**
5: $\lambda = \text{argmin} \, |A_j - cntd^k|_2^2$
6: **end for**
7: $cltr^\lambda = [cltr^\lambda A_j]$
8: **end for**
9: $d \in \mathcal{R}^{M \times 1} = 0$
10: **for** $k \leftarrow 1$ to K **do**
11: **for** $j \leftarrow 1$ to $size(cltr^k)$ **do**
12: $d = d + cltr_j^k$
13: **end for**
14: $cntd^k = d/size(cltr^k)$
15: **end for**
16: **end for**

2 K-means Algorithm

The K-means algorithm is shown in algorithm 1. The data set $A \in \mathcal{R}^{M \times N}$ is taken as input to the K-means algorithm. The algorithm also takes K and I as inputs where K represents the number of clusters and I represents the total number of iterations. This iterative algorithm takes I number of iterations to segregate the data samples in K clusters. Initially the centroids are set as $cntd^k = A_k$ for $k = 1$ to K. Here, A_j represents the j^{th} column of A and $cntd^k$ represents the k^{th} centroid.

The next step is to find the similarities between the centroids and the columns of A. Similarity can be computed by the techniques such as correlation, euclidean distance and Manhattan distance. In this work, euclidean distance is used to find the similarity between two vectors. The euclidean distance between two vectors of length M can be computed as

$$d(x, y) = \sqrt{(x_1 - x_2)^2 + (x_1 - x_2)^2 + ... + (x_M - x_M)^2} \tag{1}$$

The arithmetic operations involved in computing $d(x, y)$ are subtraction, squaring, addition and square root. The square root operation is complex and thus its hardware complexity is high. In this work, the square root operation is avoided and thus partial euclidean distance is computed. The step 5 in algorithm 1 evaluates this partial euclidean distance. The parameter λ represents the index of the column which gives minimum euclidean distance.

Once the similar columns are identified, the cluster formation is done in step 7. The parameter λ can takes value from 1 to K. Thus there are K clusters possible. The steps 9-15 of the algorithm 1 are for the averaging step. The elements of each cluster are accumulated and result of accumulation is divided by the size of that cluster. After this step, new values of the centroids are computed. The above mentioned steps are repeated for maximum I number of iterations.

3 Proposed Architecture

The proposed data path architecture of the K-means algorithm is shown in Figure 1. In this work, the K-means algorithm is implemented for parameters $M = 2$, $N = 8$ and $K = 3$. Thus a prototype of K-means algorithm is implemented here but can be adopted for higher data samples. The K-means algorithm has mainly three steps viz.,inner product computing, sorting and average computation. These three steps are executed sequentially. In this work, a hardware efficient architecture is proposed which with moderate timing complexity. All the major blocks are explained below.

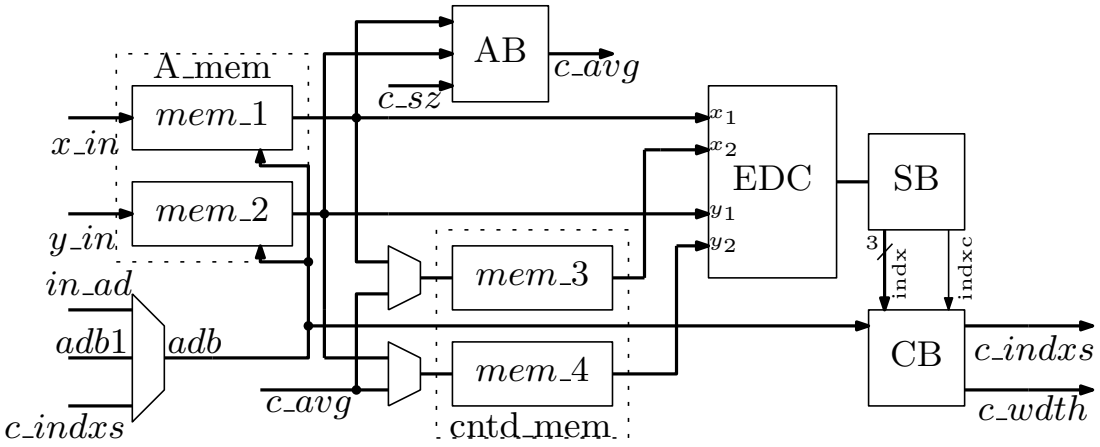

Figure 1: Proposed Architecture of the K-means Algorithm

3.1 Data Acquisition and Initialization

The input data matrix (A) can be pre-stored or can be acquired in real time. This storing of matrix A is done by A_mem which is a bank of m memory elements. Each memory element is realized using dual port rams. Port A is used for writing and port B is used for reading. In K-means algorithm K elements are randomly chosen. Initially K data samples from the A_mem are read and written to C_mem. This initial indices can be stored in a ROM or generated randomly. This proposed scheme is shown in Figure 1.

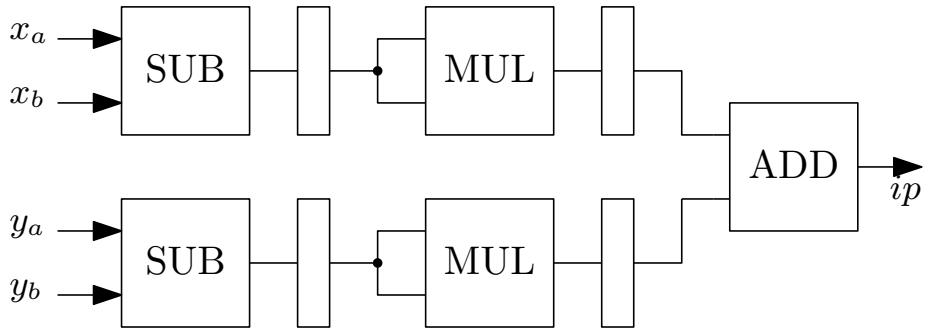

Figure 2: Proposed Architecture of EDC

3.2 Euclidean Distance Calculator

Euclidean Distance Calculator (EDC) block computes the partial euclidean distance between the centroids and the data samples. The proposed architecture of the EDC is shown in Figure 2. The initial blocks are subtractors and then squaring operation is performed. The square

blocks consumes less resources than the multipliers. The outputs of the multipliers are input to an adder tree.

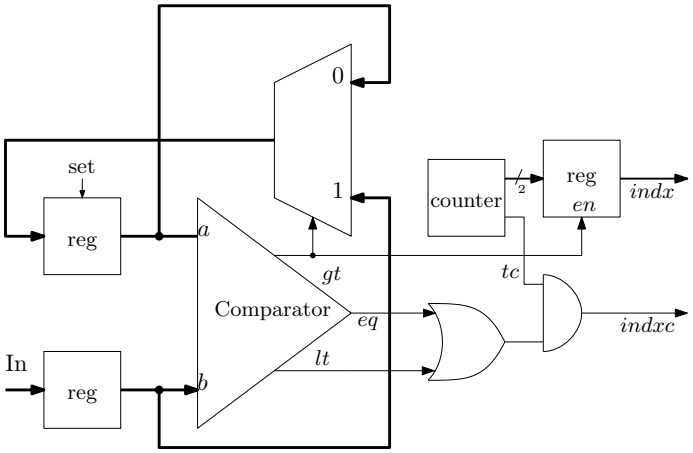

Figure 3: Proposed Architecture of SB

3.3 Sorting

The inner products computed by the EDC block are fed to a Sorting Block (SB). The SB sorts a serial stream of data and find minimum of it. The architecture of the SB is shown in Figure 3. Initially input A of the comparator is set and the signal gt selects the input A. SB outputs the index value *indx* from the N for which the value of *ip* is minimum. Also, outputs the control signal *indxc*.

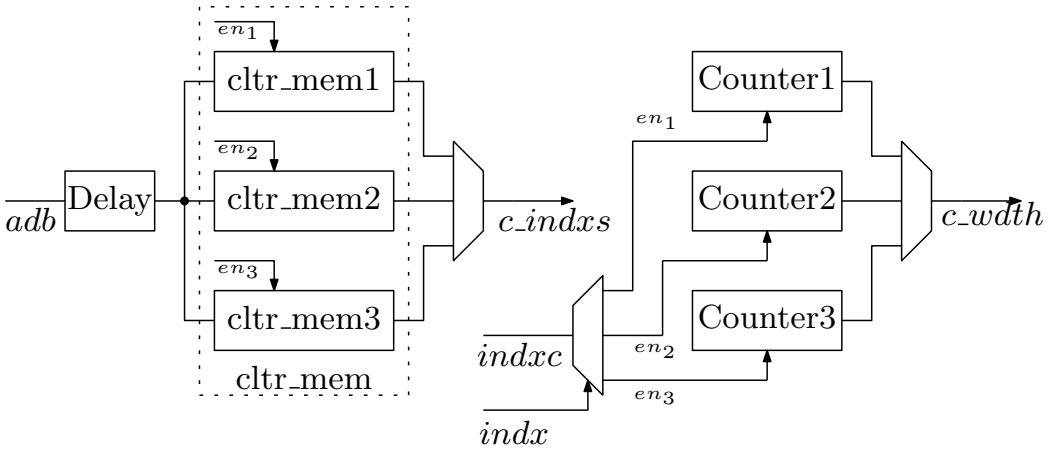

Figure 4: Proposed Architecture of CB

3.4 Cluster Block

The Cluster Block (CB) basically counts the number of data samples chosen in a cluster. Its architecture is shown in Figure 4. It receives *indxc* and *indx* from SB. CB generates K control signals en_1, en_2, en_3. These signals are used to increment K counters. Each counter holds the size of the respective clusters after the sorting operation. Simultaneously these control signals write the value of addresses (*adb*) in respective memory elements for which they are generated. Here K memory elements are used to store the indices belong to each cluster.

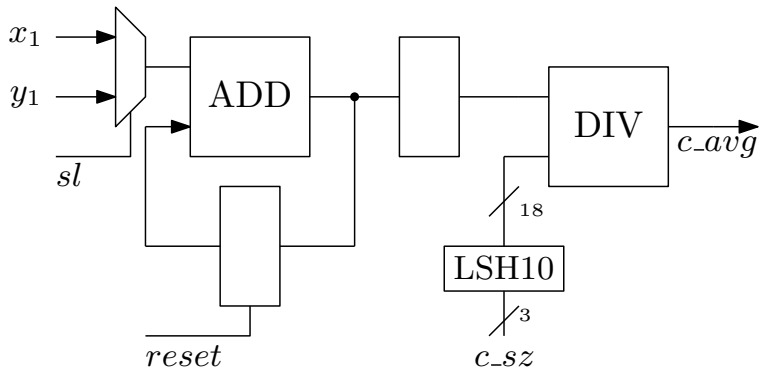

Figure 5: Proposed Architecture of AB

Table 1: Design Performance

Design Metrics	Values
Device	(xc7a100t)
Parameter (M, N, K, I)	$2, 8, 3, 3$
Slice Registers	557
Slice LUTs	527
Occupied Slices	268
DSP Blocks	2
Maximum Frequency	250
Dynamic Power (μW)	57

3.5 Average Block

The last step of the K-means algorithm is averaging. An Average Block (AB) is proposed here to compute the average of the elements selected in each cluster. The proposed architecture of the AB is shown in Figure 5. The averaging operation is done serially due to the high complexity involved in division operation. Here for the simple prototype, first averaging operation is done for x-elements and the for y-elements. A single divider is used here. The divider is non-restoring algorithm based radix-2 divider [2]. The LSH10 block is 10-bit wired left shift block.

4 Design Performance

In this work, a basic prototype of K-means algorithm is implemented for parameters $M = 2$, $N = 8$ and $K = 3$. The proposed architecture implemented on NEXYS DDR2 artix7 FPGA device (xc7a100t-3csg324). Here, 18-bit fixed point data width is chosen for implementation where 10-bit is reserved for fractional part. The design performance of the proposed architecture is shown in Table 1.

4.1 Hardware Complexity

Analysis of the hardware complexity of the proposed architecture is carried out to estimate the usage of memory elements and the major hardware blocks. The estimation of the memory blocks is shown in Table 2 and the estimation of the major hardware elements is shown in Table 3. Here k and n represents the number of bits to represent K and N respectively. This proposed architecture is designed such a way to reduce hardware resources. A single accumulation unit and a single divider block is used in this design.

Table 2: Estimation of memory elements.

Memory Elements	Word Per Cycle		Size
	Write	Read	
A_mem	M	M	$M \times N \times 18$
$cntd\_mem$	M	M	$M \times K \times k$
$cltr\_mem$	K	K	$K \times N \times k$
$indx\_mem$	1	1	$K \times n$

Table 3: Estimation of hardware complexity

Blocks	Multiplier	Comparator	Add_sub	Divider
EDC	K	0	$2M-1$	0
SB	0	1	1	0
AB	0	0	1	1

4.2 Timing Complexity

In this section, the timing complexity of the proposed architecture is discussed. It is important to derive an expression to estimate the overall execution time of the architecture. Initially K clock cycles are taken to load the initial centroids to the $cntd\_mem$ from A_mem. Then the inner product computation starts. The total time taken to complete the inner product computation stage is $(K+1)\times N+l_{ip}$ clock cycles. Here l_{ip} is the latency of the EDC block which is 3 clock cycles here. Tough the sorting operation is carried out in parallel to the euclidean computation, two clock cycles are wasted to start the next step. The next step is perform the averaging operation. The timing complexity of the averaging operation is $2K + N + l_{dv}$ clock cycles. Here l_dv is the latency of the divider block and the value of l_{dv} is 30. The total timing complexity for this prototype design is $(3K + N(2 + K) + l_{ip} + l_{dv} + 2)I$ where I is the total number of iterations. Total number of clock cycles are 246 cycles for $I = 3$. Thus total execution time is 984 μs for frequency of 250 MHz.

5 Conclusion

In this work, a novel architecture is proposed to implement the K-means algorithm. A prototype model is implemented on NEXYS4 DDR2 FPGA device. This design is scalable and can be adopted for any size of matrix A. The proposed architecture is hardware efficient with moderate execution time. A single divider is used in this design to reduce the hardware resources. A detail analysis of timing and hardware complexity is presented in this manuscript.

References

[1] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[2] S. Roy, *Division Algorithms*, no. online. [Online]. Available: "https://digitalsystemdesign.in/signed-array-divider/"