Design Trade-offs in Configurable FPGA Architectures for K-Means Clustering

Alexandru AMARICAI

University Politehnica Timisoara, Vasile Parvan Blvd, Nr. 2, Timisoara, Romania.

email: alexandru.amaricai@cs.upt.ro

Abstract: K-Means clustering is a popular technique for data partitioning, frequently used in data mining. The simple control flow, and high degree of parallelism, makes it a good candidate for FPGA acceleration. We propose a highly configurable architecture, based on Euclidean distance computation. It can be tuned by the following parameters: number of dimensions, dimension width, dimension based parallelism degree, number of centroids and centroid based parallelism degree. We study their impact on different K-Means components, such as the distance computation, distance comparison, accumulation, division, or the memory modules within the accelerator. Furthermore, for the aforementioned parameters we investigate the performance/cost trade-offs of the proposed K-Means accelerator implementation.

Keywords: K-Means clustering, FPGA, parallelization.

1. Introduction

K-Means clustering represent one of the widely used unsupervised machine learning algorithms. It is highly used in a wide variety of applications, such as data mining, radar tracking, image colour quantizations, spectral clustering of images, etc- [0],[5],[6],[7],[8]. The algorithm relies on grouping a set of multi-dimensional data points in clusters, which are defined by the cluster centroid. Two main features for this algorithm are represented by its simplicity and its high degree of parallelism at different granularity levels – [0],[9]. The first feature is due to both the way points are assigned to a cluster, as well as the computation of the new set of centroids. Assignment of a data point to a cluster is performed by computing the distances between the point and all of the centroids; the point is assigned to the minimum distance centroid. A new centroid is computed as the mean of all the points assigned to it. This need for parallel computations makes, K-Means clustering a good candidate for hardware acceleration, especially for FPGA devices.

In this paper, we propose a highly parameterizable accelerator for K-Means clustering. The main contributions for the proposed architecture are the following:

- Very high versatility for a wide variety of clustering applications, as the proposed architecture can be configured for the following parameters: dimension data width, number of dimensions of the points in the data set, and number of centroids.
- High flexibility with respect to different cost and performance trade-offs, as the proposed architecture can be configured for different dimension based parallelism degree and centroid based parallelism degree. To the best of our knowledge, this work is the first approach which investigates the parallelism for the K-Means clustering at different architecture levels and granularities.

The paper is organized as follows: Section 2 presents the K-Means algorithm and the related hardware approaches to accelerate it; the proposed architecture is described in Section 3; synthesis results for different configurations are discussed in Section 4; last section is for concluding remarks.

2. Hardware Acceleration of K-Means Clustering

2.1. K-Means Clustering

K-Means algorithm is a heuristic iterative algorithm which groups a set of data points $\{P_i\}$ in a number of n_c centroids, defined by their centroids $\{C_j\}$. The data is multi-dimensional, having n_d dimensions, each dimension being represented on a number of w bits. The algorithm performs several iterations, each iteration consisting of the following steps [0],[8]:

1. Distance computation – for each point in the data set, the distance between it and all of the centroids is computed;

2. Distance comparison – for each point in the data set, the minimum distance to a centroid is determined; the point is allocated to the centroid with the minimum distance;
3. Point accumulation – for each centroid, the sum and the number of all points allocated to it is computed;

Computation of new centroids – the new centroids are obtained by the arithmetic mean of all points allocated to it (division between the sum and the number of points);

Regarding distance computation, Euclidean or Manhattan distance can be used. The Euclidean distance - $\sqrt{\sum_{k=p}^{nd}(c_k - p_k)^2}$where c_k and p_k represent the k-th dimension of the point P_i, and centroid C_j - has more computational complexity [2]; nonetheless, it provides more accurate results and faster convergence than Manhattan. The Manhattan distance - $\sum_{k=0}^{n_d}|c_k - p_k|$ - is simpler to compute; however, it lacks the accuracy and convergence of the Euclidean distance.

We define the dimensional parallelism degree p_d, which represents the number of difference and square operations within the distance computation performed in parallel. We define the centroid parallelism degree, denoted by p_c, as the number of distances between one point and centroids computed in parallel.

The above algorithm, known as the Lloyd algorithm, is the basic K-Means implementation. Improvements for this algorithm relied mainly on the reduction of distance computations, using triangle inequalities – [2],[6], or on performing the algorithm on tree based data structure – [9]. The main drawbacks of these approaches are represented by their complex control flow, large memories, as well as large overhead due to some preliminary computations, such as distance computation between centroids or centroid sorting. Therefore, their FPGA implementation may require increased area and power consumption overhead.

2.2. FPGA Acceleration of K-Means

Several K-Means clustering accelerators for [6],[9], use algorithmic improvements for clustering, such as the triangle inequality – [6], or the tree based data structures – [9]. Approaches in [0],[4],[5],[8] implement the classic Lloyd algorithm. The approach in [8] performs the distance computation, distance comparison and point accumulation using integer numbers, while the division is performed in floating point. It uses Manhattan distance, using a maximum centroid based parallelism degree. The approach in [0] for micro-array data clustering uses Manhattan distance as well, and maximum centroid based parallelism degree. Furthermore, the imple-menttation is based on the storage of data points within the internal BRAM memories. Therefore, this approach lacks scalability. The approach in [4] propose a parameterizable design, which can be tuned by the number of centroids, number of dimensions and data size. It implements Manhattan distance, while the centroid and dimension degree parallelism is maximum. The approach in [5] aims at maximizing performance, by utilizing maximum parallelism at both centroid and dimension degree, as well as maximizing the working frequency. It uses Manhattan distance, while the data set is stored in the internal BRAM memory; thus, this approach may lack scalability when dealing with large data sets. As it can be observed, a high parallelism degree is usually used with the Manhattan distance, due to the lower computation cost.

3. Parametrizable K-Means Architecture

3.1. Architecture

The proposed architecture is depicted in Fig. 1. It presents the main components specific to an accelerator dedicated to Lloyds K-Means algorithm. It consists of a distance computation unit, distance comparison module, the point accumulator unit, and dividers. Centroids are stored into a dedicated memory, implemented using BRAM blocks. We are targeting large data sets - several tens of MB or more of data -, which cannot be stored into the internal FPGA memory. The accelerator interfaces with the rest of the system through conventional bus interfaces, such as AXI. FIFO buffers are used for data points for communication with the bus interface. Therefore, in order to fetch the data points to the FPGA based accelerators, writing into the FIFO buffers is performed.

The number of FIFO buffers is equal to the dimension based parallelism degree p_d.

The overall architecture has the following parameters: number of dimensions n_d , number of centroids n_c, dimension size w, dimension based parallelism degree p_d, centroid based parallelism degree p_c, and the depth of the input FIFO buffer d_f . All these parameters - Verilog design parameters – are applied to the top level module of the proposed architecture.

3.2. Distance computation

In the proposed design, we have used the square of the Euclidean distance - $\sum_{k=0}^{n_d}(c_k - p_k)^2$. The distance between one point and one centroid is computed on full precision - $2 * w + n_d$bits. An inferred based multiplier is used for squaring in order to increase the generality of the proposed unit. A number of p_d difference and square units are employed. An accumulator is employed in order to derive the distance. A number of n_d/p_d accumulations are performed.

A number of p_c distance computation units may be used, which allows the parallel calculation of the Euclidean distance between one point and p_c centroids. In order to compute the distance between one point and all centroids, n_c/p_c iteration have to be performed. After reading it from the FIFO buffers, the point data is stored into a register based buffer. A local control unit for the distance computation is employed. It generates the appropriate read addresses for the centroid memories, as well as the read enable signals for the input FIFO buffers.

3.3. Distance comparison

We have employed an array comparator for distance comparison, due to the fact that is suited for parametrized implementations. Furthermore, the array comparator has a smaller cost with respect to a tree comparator. However, when high performance is required, comparison trees may be employed. The comparator has $p_c + 1$ distance inputs, when $p_c < n_c$, and n_c distance inputs, for fully parallel design. The comparison is performed in n_c/p_c.The output of this module is represented by the index of the centroid with the minimum distance.

3.4. Point accumulation

The point accumulation unit is composed of a point accumulation memory, p_d adders and n_c counters. The accumulation of the w -bit points corresponding to a centroid is performed on $2 * w$ bits, in order to have enough precision. The point accumulation memory word has the size of $2 * w * p_d$, while the depth of this memory is $n_c * n_d/p_d$. The addition corresponding to a centroid is performed in n_d/p_dclock cycles. The index of the minimum distance is used in order to generate the appropriate read/write addresses for the point accumulation memory, as well as to increment the corresponding counter.

3.5. Division

Division is performed at the end of the iteration. The required number of individual divisions is $n_c * n_d$. In the vast majority of applications, the number of centroids is several orders of magnitude lower with respect to the number of points which require clusterization. Thus, the impact of division on the overall performance of the K-Means clustering may become negligible. Therefore, we have opted for a low cost integer sequential divider, based on the non-restoring division algorithm. The number of dividers used is equal to the dimension parallelism degree p_d. Several rounds of division - n_d / p_a are performed for each centroid. As the centroid memory word contains p_c* p_d w-bit data, while at each division round a number of p_d w-bit data are obtained, a shift register is mployed

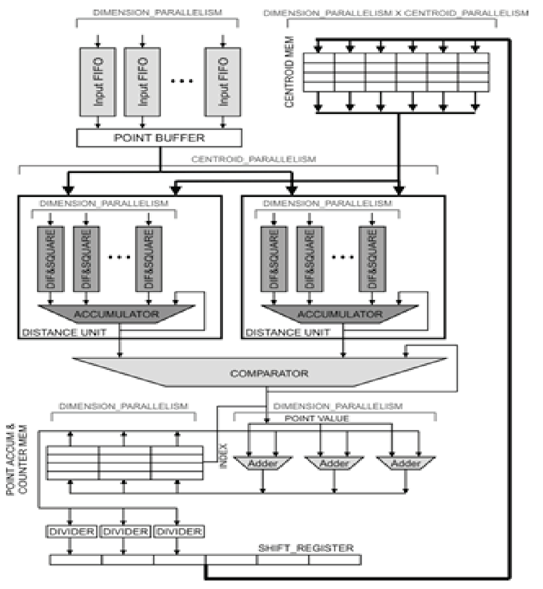


Figure 1. Architecture of proposed K-Means FPGA Accelerator

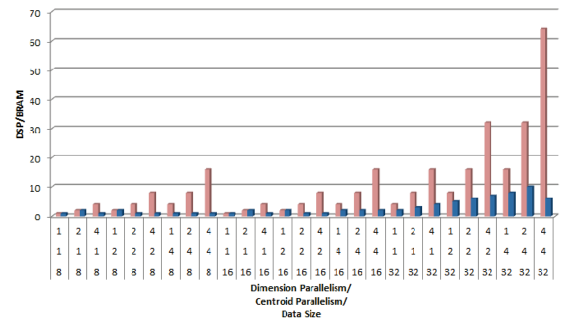
before the centroid memory write. The shift register has as input the outputs of the p_d divisions, and outputs the p_c* p_d *w*-bit centroid memory word.

3.6. Centroid memory

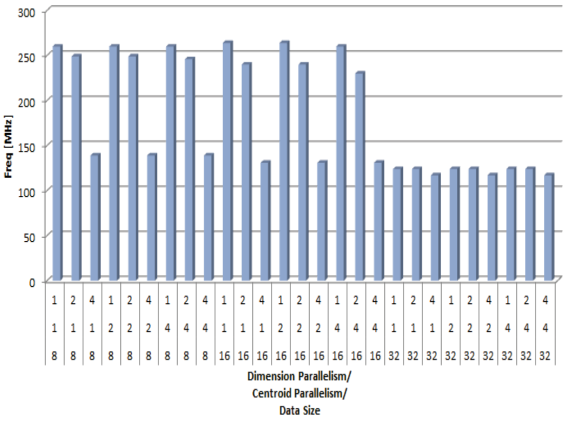
The centroids are stored in the internal memory resources of the FPGA. The centroid memory is organized into $p_c * p_d$ memory banks, each of w-bit width. The depth of each memory bank is $(n_c * n_d)/(p_c * p_d)$ being a para-meterizable module, the centroid memory is inferred. Depending on the depth of the memory banks, either BRAM memory is synthesized - for low parallelism degrees -, or the memory is implemented using distributed RAM. The memory banks can be accessed either in an independent manner or in block. The former type of accessing is used when the centroid memory is accessed from the IO interface; these kinds of accesses are used to write the initial centroids or to read the centroid data after an iteration. The block type of access is used when the centroids are read for distance computation or the new centroids are written.

4. Synthesis Results

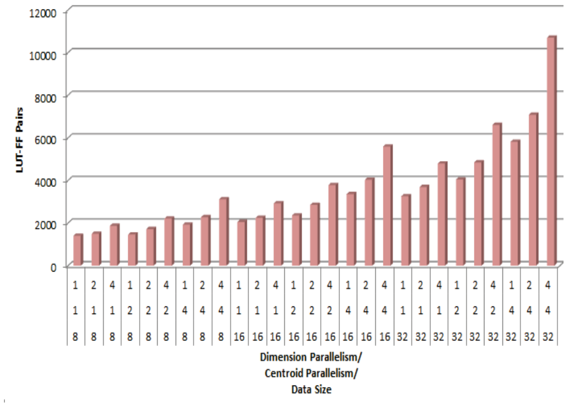
Synthesis results for the proposed architecture for Xilinx Zynq7020 device, using Xilinx ISE 14.7, speed grade -2, are presented in Figure 2. Results for a data sets consisting of 8-bit points, 16-bit points and 32-bit points, with 8 dimensions and a number of 16 centroids have been obtained. The considered parallelism degrees for both centroids and dimension have been 1, 2 and 4. Regarding the parallelism degrees, the only restriction is that n_c and n_d are integer multiples of p_c, and p_d respectively. For n_c = 16 possible values of p_c are 1, 2, 4, 8, and 16. Regarding the data size, 8-bit dimensions are frequent in image clusterization - 6, while 16-bit data is common to sensor based data clusterizations (outputs of a 16-bitare integer multiples of p_c, and p_d respectively. For n_c = 16 possible values of p_c are 1, 2, 4, 8,and 16. Regarding the data size, 8-bit dimensions are frequent in image clusterization - 6, while 16-bit data is common to sensor based data clusterizations (outputs of a 16-bit ADC). The considered input FIFO buffers have a depth of 32. For this small depth, a distributed RAM implementation is more suited. When large FIFOs are required - with a depth of more than 256 -, the dedicated BRAM based FIFO primitives/macros can be used. Regarding the synthesis process properties, automatic inference of both the DSP blocks and BRAM modules has been used. Although implementation post place-and-route results provide the most realistic estimates, we have opted to present the synthesis ones, due to the very high number of configurations - 27. Verification has been performed in SystemVerilog, using co-simulation with a golden C model. Cost results suggest that dimension based parallelism degree has a slightly stronger influence with respect to that of centroid based parallelism degree. The K-Means accelerator



(a)



(b)



(c)

Figure 2. Frequency estimates (a)/ Slice based cost (b)/ DSP and BRAM usage (c)

with p_d =4, p_c =1 has a slight increase cost in terms of logic and BRAM with respect to a configuration with p_d =1, p_c =4. Regarding the BRAM usage, increasing the parallelism degrees may not lead to an increase in the used BRAM blocks. This is due to the fact that for large parallelism degrees, employing distributed RAM would be more efficient, due to the large words and low depths of the centroids memory.

Regarding the working frequency, the critical path can be found either in the square component, or in the distance comparison module. For 8 and 16-bit data, square units use only 1 DSP per operation. In this case, the critical path is in the array comparator. For 32 bits operands, the square unit is implemented using 2 DSP blocks. For small p_c, the critical path is in the square unit. For large p_c values, the critical path is in the array comparator.

Figure 3 presents performance estimates in terms of clock cycles, as well as processing time. The estimates are for data sets consisting of 16384 points, with 4 dimensions and 8 centroids. The results are simulation based, for 16 bit data size. Regarding fetching the data points, only one write transaction for writing one dimension (4 transactions for 1 point transfer) has been considered. Increasing the parallelism will lead to an almost linear decrease in the required number of clock cycles, due to the following: (i) increase in processing capacity, (ii) increase of the size of

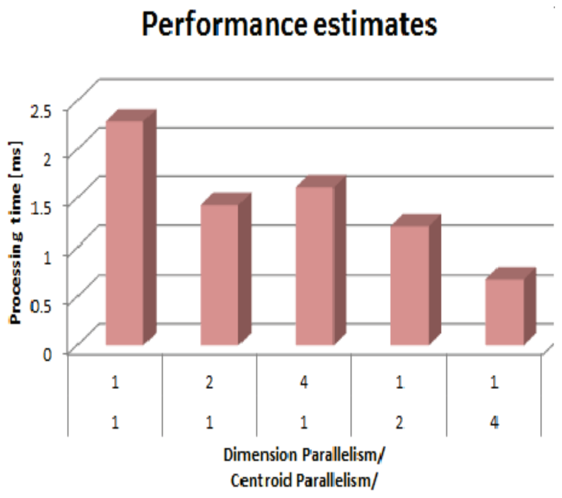


Figure 3. Performance estimates

the input FIFO buffers, which may lead to fewer stall cycles for writing the points. Results indicate that increasing the dimension based parallelism degree leads to a higher performance with respect to the same centroid parallelism degree. This is due both to the increased clock frequency and the reduced number of clock cycles. Furthermore, we observe that the low frequency associated with pc increase may lead to lower performance with respect to a small dimension parallelism degree: an unit with p_d = 2 and p_c = 1 has smaller processing times with respect to an unit with p_d = 1 and p_d = 4.

5. Conclusions

In this paper, we proposed a highly parameterizable architecture for the Lloyds K-Means clustering algorithm. The proposed architecture can be tuned according to the data width, number of dimensions, number of centroids, dimension based parallelism degree and centroid based parallelism degree. We analyze the impact of these parameters, with special emphasis on the two levels of parallelism in the K-Means unit, on the cost, frequency and performance of the proposed architecture. Synthesis results have indicated that increasing the dimension based parallelism degree leads to slightly higher cost with respect to an increase in the centroid based parallelism degree. This is because of the higher clock frequency and the lower number of clock cycles required. Future work will consist in the integration of the developed accelerator in complex systems-on-chip with dynamic reconfiguration capabilities.

Acknowledgements

This paper has been supported by the CHIST-ERA DIVIDEND research project.

REFERENCES

1. Choi,Y.M., So, H.K., (2014) Map reduce processing of K-Means algorithm with FPGA based computer cluster, *Proceedings 25th International Conference on Application-Specific Systems, Architectures and Processors (ASAP)*, pp. 9-16.

2. Elkan C.(2003) Using the triangle inequality to accelerate the K-Means, *Proceedings 20th International Conference on Machine Learning (ICML),* pp. 362-365.

3. Hussain, H. M., Benkrid, K., K. H. Seker, A. T. Erdogan (2011), FPGA implementation of K-Means algorithm for bioinformatics application: An accelerated approach to clustering microarray data, *Proceedings 2011 NASA/ESA Conference on Adaptive Hardware and Systems (AHS),* pp. 248-255.

4. Hussain, H.M.,& Benkrid, K., Erdogan, A. T., Seker, H. (2011) Highly parameterized K-Means Clustering on FPGA: Comparative results with GPPs and GPUs, *Proceedings 2011 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pp. 248-255.

5. Kutty, J.S.S., Boussaid, F., Amira, A. (2013) A high speed configurable FPGA architecture for K-Means clustering, (2013) *Proc. 2013 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1801-1804.

6. Lin, Z., Lo, C., Chow, P., (2012) Application K-means implementation on FPGA for high-dimensional data using triangle, *Proceedings 22nd International Conference on Field Programmable Logic and Applications*, pp. 437-442.

7. Luca, R. (2016) Clustering-based Human Locomotion Parameters for Motion Type Classification Studies in Informatics and Control, *25*(3), pp. 353-362.

8. Wang, W., Lesser, M., (2007) K-means clustering for multispectral images using floating-point divide, *Proceedings 15th Annual IEEE Symp. On Field Programmable Custom Computing Machines (FCCM)*, pp. 151-162.

9. Winterstein, F., Bayliss S., Constantinides, G. (2013) High-level synthesis of dynamic data structures: A case study using Vivado HLS *Proceedings 2013 International Conference on Field Programmable Technology (FPT)*, pp. 362-365.

10. Winterstein, F., Bayliss S., Constantinides, G. (2013) FPGA based K-Means clustering using tree based data structures, *Proceedings 23rd International Conference on Field Programmable Logic and Applications*, pp. 1-6.