Communications on Applied Electronics (CAE) – ISSN : 2394-4714
Foundation of Computer Science FCS, New York, USA
Volume 7– No. 30, August 2019 – www.caeaccess.org

FPGA Implementation of Forgy's K-Means Clustering for Real Time Image Analysis

Anuradha M. G.
ATME college of Engineering, Mysore
JSS Academy of Technical Education, Bengaluru

Basavaraj L.
ATME college of Engineering
Bannur Road, Mysore

ABSTRACT
Partitioning the image into meaningful groups is one of the major task in image analysis which can be achieved using the unsupervised clustering algorithm. K-means algorithm is one of the popular unsupervised clustering algorithm. The K-means algorithm is time-consuming and requires intensive computation for a large data set as the input is compared with all the centroids. Also, the data needs to be stored internally due to iterative re-assignment process. An architecture to enhance the speed of clustering operation using minimal hardware for K-means clustering without any internal storage is proposed and implemented using Virtex 6 FPGA. A new methodology is proposed to reduce the distance computation. The performance of the architecture is 203fps for a grayscale image size of 256X256 and 102fps for a grayscale image of size 512X512. This shows that the proposed architecture can be used for real time image segmentation.

Keywords
Clustering, FPGA, Image segmentation, K-Means, Machine learning.

1. INTRODUCTION
Clustering is an unsupervised clustering algorithm where the data can be grouped together depending on the similarity. The data that are similar are put in the same group. One of the popular clustering algorithm is the K-means clustering algorithm [1] that is used in many fields like machine learning [2], data mining [3] and multimedia communication [4].

The software implementation of K-Means algorithm was unable to meet the timing requirement of the systems. To meet the increased demand and due to its simplicity, many hardware architectures are developed to accelerate the clustering operation [5]-[15]. The hardware specifications of these works vary because of the different target applications.

Filho et al. propose a software/hardware co-design technique for K-Means clustering [5], which is used for clustering the hyper spectral images. The distance calculation and class selection for K-Means algorithm was implemented in software as they are computational intensive and require more hardware resource.

Maruyama [6] proposes an FPGA implémentation of K-Means clustering for color images where 4 pixels are processed in parallel. To reduce the distance computation from each point to the cluster centers, KD tree filtering algorithm is used in [7]. T.-W.Chen and S.-Y. Chien propose hardware architecture of K-Means clustering where bandwidth adaptive mechanism using 5 parallel modes for different vector dimensions is proposed to effectively use the hardware [8]. The divisive hierarchical clustering algorithm with K-Means clustering in proposed in [9] to handle high clusters. Hardware architecture which address the initial centroid selection and acceleration of the clustering process are proposed in [10] - [15]

In the previous work [16], Online K-Means clustering was developed for handling vector dimension up to 8 with maximum of 16 clusters. The online architecture developed provide the clustering result in single iteration with less memory overhead for storing the input vectors but however, the clustering quality can be improvised by Forgy's K-means due to iterative assignments.

The architecture developed till now for clustering an image compares each pixel value with the centroid and the data is placed in the cluster with nearest centroid. In papers [6] and [7], few boundary points are selected after the first iteration in K-Means to reduce the computation time. In an image, each and every data is compared with all the cluster centroid and hence the computation becomes intensive as the number of distance calculator required is proportional to the number of data N. If fixed number of distance calculators are used, then the number of iterations to compare the data with the cluster centroid increases. For a gray scale image and for a 24 bit full color image, the intensity level will have values only from 0 to 255. Hence the data comparison with the centroid in an image will have many repeated calculations as the intensity levels in an image is repetitive. Hence to reduce the computations and to speed up the operation, the proposed hardware architecture computes the distance only for each intensity value ranging from 0 to 255. Initially, when the input image is read, the image pixel intensity is compared with 256 intensity values and if equal, then corresponding intensity register is incremented. Hence 256 registers only are required to store the number of pixels having the intensity level from 0 to 255 irrespective of the image size.

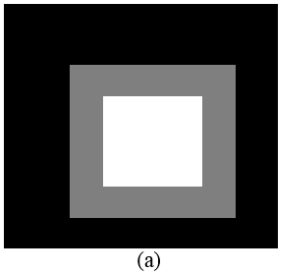
2. K-MEANS CLUSTERING AND METHODOLOGY PROPOSED
The K-means clustering is an iterative clustering algorithm that groups the data into K sub-groups. The number of subgroups K called clusters is specified prior.

The first step in K-means clustering is to select K inputs and assign that as cluster center or centroids. Each input pixel is read and is assigned to the nearest centroid.by computing the distance between the input pixel and all the centroids. After reading all the inputs and assigning to the nearest centroid, the centroid is re-computed as mean of the cluster. The process is repeated till the centroids do not change.

The assignment and re-assignment of the data to the nearest centroid needs each input data to be compared with all the K centroids. If the data size is large, then the computational complexity involved in comparing the data with all the centroids will be more. For an image of size NXN, all the N^2

pixels need to be compared with K centroids. The comparison of the data with the nearest centroid is computed as either the Euclidean or Manhattan distance between the data and the centroid. For large data set with parallel computation facility, the distance calculators required to compute the distance between the input and the centroids will be more.

A methodology is here proposed to reduce the computational complexity. In a 24 bit RGB image or an 8 bit gray scale image, the intensity values are in the range of 0 in 255. Hence in an image of size NXN, all the N^2 pixels will have the values only from 0 to 255. These intensity values only will be repeated in an image. Hence instead of comparing each and every data with the centroids, only the 256 intensity values will be compared with the centroids. To understand the concept, consider the example image shown in Fig 1. Let the intensity values in an image vary in the range 0 to 3 assuming to be 2 bit gray scale value.



(a)

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0
0	0	1	3	3	3	1	0
0	0	1	3	3	3	1	0
0	0	1	3	3	3	1	0
0	0	1	1	1	1	1	0
0	0	0	0	0	0	0	0

(b)

Fig 1: (a) 8X8 2 bit gray scale image (b) Intensity values of the 8X8 image

The image shown in Fig 1a is of size 8X8. If K-means algorithm is applied to an image shown Fig 1a., then all the 64 pixels in an 8X8 image needs to compute the Euclidean or Manhattan distance between each of the centroids. Hence for an 8X8 image, the distance should be calculated 64 times with each centroid. If the image is a two bit gray scale image, then the image pixels can have the value only from 0 to 3 as seen in Fig 1b. For the given 8X8 image, the intensity value '0' appear 39 times and hence in original K-means clustering algorithm, the intensity value '0' computes the distance with the all the K centroids 39 times. Similarly, in the given image, the intensity value '1' appear 16 times and intensity value '3' appear 9 times. Hence distance between intensity value '1' and each centroid is carried out 16 times and distance between intensity value '3' and each centroid is carried out 9 times. As seen, the same distance calculation is done (38*K) times redundantly for an intensity value black. Similar redundant calculations are done for other intensity values in an image.

To reduce the computational efforts, the distance between intensity values and each centroid is computed only once. Also the count of the pixels having the same intensity value is computed which is required to determine the new centroid. Hence instead of comparing each pixel with the centroid, only intensity values of the pixel can be compared with the centroids. Hence in the proposed method, for an 8 bit gray scale image, the distance computation between 256 intensity values and each of centroid is carried out irrespective of the image size. The architecture of proposed the methodology to compute the K-means algorithm is described in the next section.

3. PROPOSED ARCHITECTURE

The proposed architecture of K-Means clustering algorithm is designed to work under the processing platform where the width of bus is 8 bits as the input is 8 bit gray scale image. The architecture uses a Manhattan distance calculator which does not require multipliers and hence the clustering operation can be executed faster. The division module realized for finding the new centroid carries out the operation in a single clock cycle which in turn speeds up the clustering operation. Also the random initialization method is used to select the initial centroid. The architecture can cluster the data up to 8 groups or up to K=8.

An overview of the proposed K-Means clustering architecture is illustrated in Fig 2, and the functionality of each module will be explained in the following subsections.

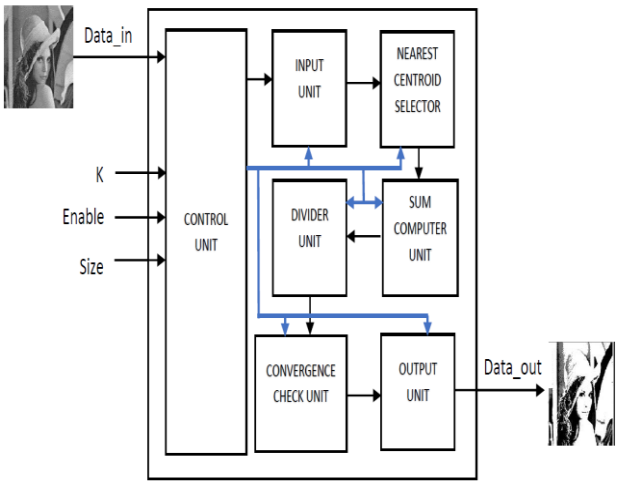


Fig 2: Proposed architecture

3.1 Input Unit

The pixel of each image is sent serially through data-in input. The input pixel is compared with 256 data values from 0 to 255 corresponding to the intensity values of an 8 bit gray scale image. There are 256 registers R_0 to R_{255} each for one intensity value which gets incremented if the input data value match the corresponding intensity value. Hence R_2 register gets incremented if the input data has an intensity value 2 and the other registers retain the previous value. The input unit is shown in Fig 3.

As seen, the comparator output act as enable signal to the adder. If enabled, then the adder adds value 1 to the existing value of the register. The comparator is used for comparing the input with a value and hence the complexity of the comparator reduces to an 8 bit AND gate as shown in Fig. 3. Hence when comparing the input with pixel value '0', the Data_in input is inverted and fed to 8 input AND gate. When comparing the pixel value '255', Data_in input is directly fed to 8 input AND gate.

Irrespective of the image size, 256 registers and comparators are needed. For an image of size 256X256, 65536 clock cycles are required for comparison of the entire image. After reading the entire image, registers R0 to R255 contains the information of the number of pixels having the intensity value from 0 to 255. The initial centroids are selected randomly from the input pixels and stored in a temporary register.

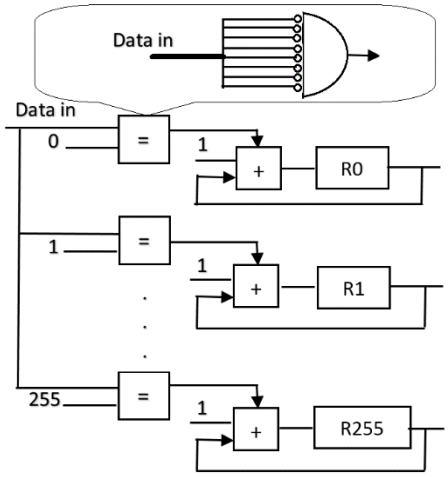


Fig 3: Input unit

3.2 Nearest centroid selector

Nearest centroid selector assigns the input pixel intensity to the nearest centroid. To compute the nearest centroid, there are fixed 256 distance calculators. The distance calculator compares the intensity values with the centroids. Manhattan distance is used for calculation of the distance which computes the absolute distance between the data and the centroid. The distance calculator that computes the distance between the input pixel data 'D' with 4 centroids C1, C2, C3 and C4 is shown in Fig 4

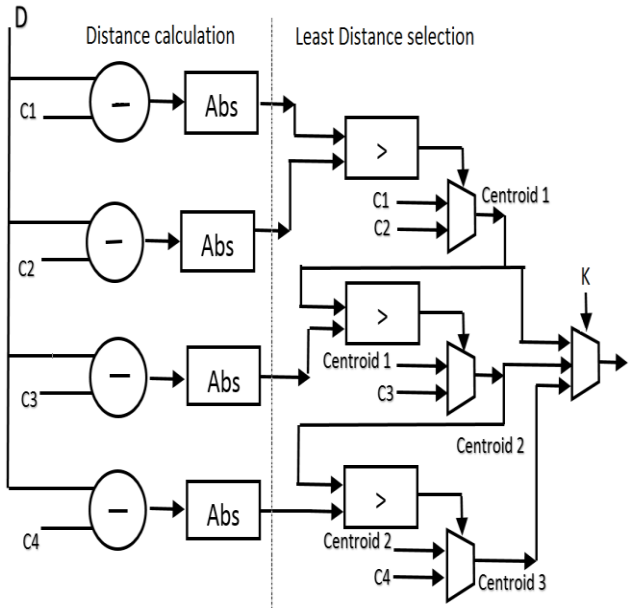


Fig. 4: Nearest centroid selector

The absolute distance between the input pixel and all the centroids is computed in the section shown as 'Distance calculation' of Fig 4. The 'Least distance section' shown in Fig 4 selects the value of the nearest centroid depending on the cluster K value.

The input pixel is subtracted with all the centroids C1, C2, C3 and C4 .The absolute value of the difference is taken to obtain the distance between the input pixel and all the centroids. The Manhattan distance is available as output from the section 'Distance calculation'. The distance calculated is used as select input for the mux for selecting the centroid that has least distance with the input.

For K=2, the least distance selection unit selects the centroid called 'Centroid 1' that has least distance with the input pixel. For K=3 , the least distance selection unit selects the centroid called 'Centroid 2' that has least distance with the input pixel and similarly when K=4, , the least distance selection unit selects the centroid called 'Centroid 3' that has least distance with the input pixel. The selected centroid is stored in the temporary register. The single nearest centroid shown in Fig 4 supports for clustering the data up to 4 clusters. The architecture supports the clustering operation up to K=8 where there are 8 subtractors, 8 absolute unit, 7 comparators and 8 multiplexers.

For each value of the input pixel from 0 to 255, the centroid nearer to the data is selected using 256 nearest centroid selectors and stored in 256 corresponding registers.

3.3 Sum Computer

The sum computer adds the pixel values nearer to the centroid. If there are four centroids, then the data points nearer to the centroid1 are added together and stored in a register called sum1. Similarly, the data points nearer to the centroid2, centroid3 and centroid4 are stored in sum2, sum3 and sum4 registers. Also the Sum computer computes the total number of data near to centroid1 to centroid4 and stores it in the register count1 to count4.

The output of each nearest centroid selector is compared with all the centroids. If the nearest centroid output is equal to centroid1, then the corresponding intensity value of a pixel will be multiplied by the total number of pixels having that intensity value in the image which is stored in register R_0 to R_{255} in the input unit and is updated in the sum1 register. Hence sum1 register contains the sum of all the pixels nearer to centroid 1.The pseudo-code for finding the sum and count and to compute new centroid is given below

Algorithm 1: To compute sum of data in a cluster and finding the new centroid

```
Initialize sum=0, count=0
for i<-0 to 255
{
 if (nearest_centroid_out (i) = centroid)
            {
             sum= sum + (i*reg(i))
            count= count + reg(i)
            }
}
New_centroid = sum/count
If New_centroid = centroid
  Done =1
Else
Done =0
```

3.4 Divider

The new centroid in K-means clustering is obtained by dividing the sum by count as seen in algorithm 1. The divider is designed to operate in a single clock cycle. The sum and count registers are designed to be 24 bit registers and hence the divisor and the dividend for the division operation is 24 bits. The pseudo code for obtaining the quotient and remainder is given below for n bit division.

Algorithm 2: To compute the new centroid

```
Initialize temp_divisor = concat {1 bit of value, divisor, (n-1) bits of value 0}
                temp_remainder = concat{(n-1) bits of value 0, dividend}
  If (dividend=0)
 {
   quotient= 0;
  }
  else
          {
        For i <  0 to (n-1)
        {
           temp_result = temp_remainder - temp_divisor;
                    if(temp_result((2n-1) - i)=1)
                    quotient(n – i) = 1'b0;
                    else
                    {
                    Quotient (n – i) = 1'b1;
                    temp_remainder = temp_result;
                    }
                    temp_divisor = temp_divisor >> 1;
                }
```

3.5 Convergence monitor and output unit

"Convergence Monitor" module examines whether the quotient obtained from the divider is equal to the previous centroid value and returns the information to the "Control Unit" module. The control unit is modelled as Finite state machine is and it checks if the centroid is converged. If so, then the information is given to the output unit else the new iteration begins again from centroid selector.

4. RESULTS AND DISCUSSION

The proposed K-Means architecture is built using Verilog HDL and tested using I-Sim simulator and MATLAB. The architecture is implemented on Xilinx Virtex 6 FPGA.

The first part in the verification is testing of the algorithm developed for image analysis. The gray scale images of various size that needs to be tested was stored in an external memory and one image pixel per clock cycle was sent to the architecture. The image after clustering was put back in an external memory for analyzing the clustered image. The input for the architecture was an 8 bit gray scale level and the output is the 8 bit gray scale value corresponding to the nearest centroid for each input pixel. Fig.5. shows the clustering result for various values of K

The architecture was tested using various standard images like Lena, Cameraman, Baboon, Ship, peppers and house image as it covers the mixture of flat regions, shading and texture in an image.

The second part is analyzing the hardware cost of the architecture. To analyze the area occupied by the proposed algorithm, the original K-means architecture that compares each data with the centroid was also designed and tested. The original K-means architecture built also had 256 fixed distance calculator to compute the distance between the input and the centroid. Hence only 256 pixels can be processed at a time. The Slice LUTs of original K-means architecture is more than double compared to the proposed architecture as seen in the Table 1. Also the hardware requirement of original K-means architecture increases with increase in the image size. Also by varying the input image size, the hardware cost of the proposed algorithm was tested. It is seen that the hardware utilization remains same as shown in Table 1 for varied image size. However, the initial comparison time varies with the image size.

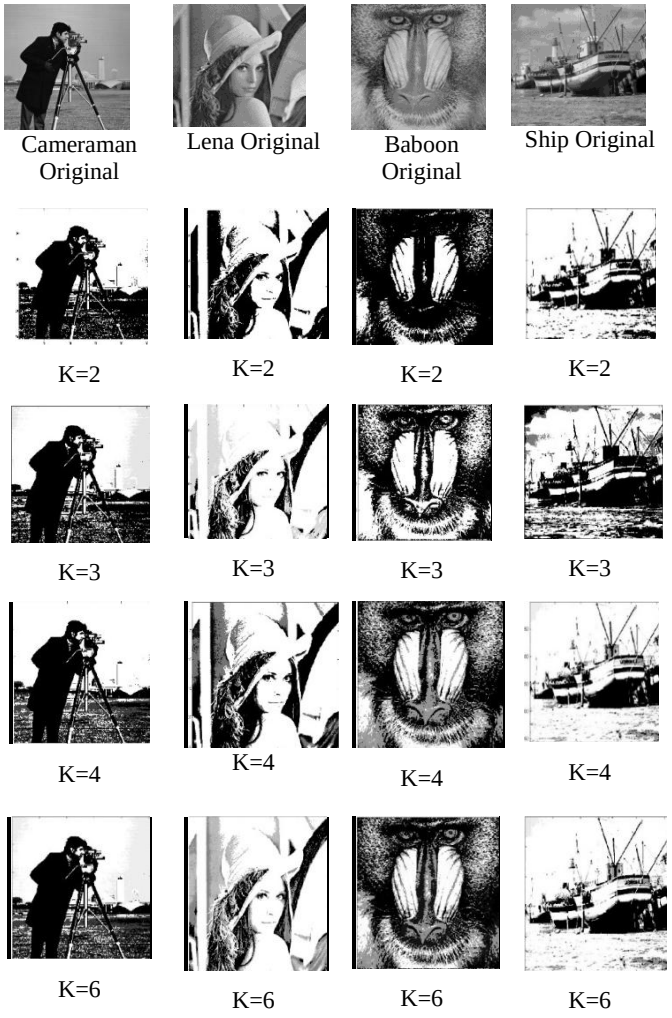


Fig 5: Clustering results for various images

Table 1: Comparison of hardware utilization of proposed work with original K-means architecture

Logic Utilization	Original K-Means Architecture	Proposed K-Means Architecture
Number of Slice Registers	6767	6813
Number of Slice LUTs	140098	41143
Number of fully used LUT-FF pairs	4615	2405
Number of DSP block	236	2

As a third step in hardware cost analysis, the cluster number K was varied in the proposed architecture and the LUTs required was analyzed. The Slice LUTs required as the K is varied is shown in Fig 6. Utilization of Slice LUTs increases almost linearly with the K because the hardware complexity of the nearest centroid calculator increases with K.

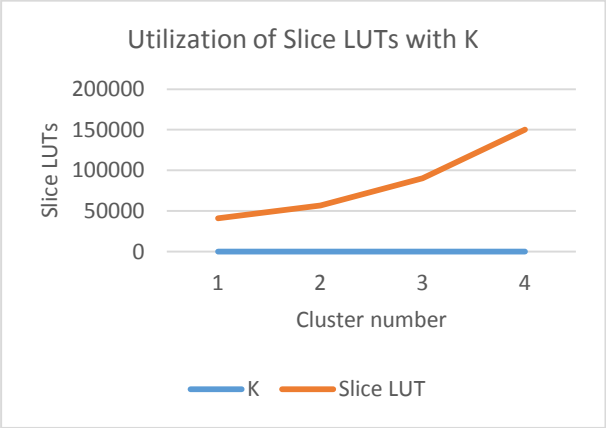


Fig 6: Utilization of Slice LUT with K varied

The third step is the analysis of the timing constraint of the architecture built. The proposed K-means architecture operate on the intensity levels of a given image irrespective of the image size. The data moves from input unit to the output unit sequentially. The intermediate modules like nearest centroid selector, sum computer, divider, convergence check unit complete their operation in a single clock cycle as all the 256 intensity levels are processed simultaneously using the parallel hardware structure. The output unit is designed using two ways. The first design of output unit is to again scan the input image and assign the determined nearest centroid for the input pixel. The second design is to store the input in the BRAM and compare all the pixels with the computed nearest centroid using parallel architecture. The second method require more hardware but can operate in a single clock cycle.

The input unit requires, 65,536 clock cycles to compare the intensity levels of all the pixels in a 256X256 sized image, 2,62,144 clock cycles to compare the intensity levels of all the pixels in a 512X512 sized image and so on. After initially comparing the image pixels in the input unit, rest all the units like nearest centroid selector, sum computer, divider, and convergence unit with output unit would take 1 clock cycle for completion. Totally, it requires 4 clock cycles for completion of 1 iteration. Even if the maximum number of iterations required is 10, then totally 40 clock cycles are required. To complete the clustering operation for 256X256 image, we require 65576 clock cycle. The architecture operates at a speed of 26.622 MHz and hence the entire image can be processed in 0.0049 sec which indicate that the architecture can process 203 frames per second. Similarly, 512X512 image can be processed in 0.0098 sec and can process 102 frames per second. This indicate that the proposed hardware architecture is well suited to work in the real time environment.

The last part of verification is comparison of the work with previous literature. This proposed architecture is compared with work of [8]. The proposed architecture performs division operation in a single clock cycle and can process more number of image frames per second.

Table 2: Comparison of the proposed architecture with related works

Specifications	T.W.Chen[8]	Proposed architecture
Implementation method	ASIC-TSMC 90 nm	Virtex 6 FPGA
Maximum cluster number	1-16	1-8
Maximum data number	2^{20}	Any image size
Clock cycles required for division operation	More than 10	Single clock cycle
Frame rate	Not specified	256X256/ 203 fps
Distance calculator	Manhattan/Euclidean	Manhattan

5. CONCLUSION

Hardware architecture of K-Means clustering algorithm is proposed. Majority of the time is consumed in K-means clustering by computing the distances between each of the K centroids and the image data pixel. Much of the distance computations are repetitive as the intensity levels in an image are repeated numbers within the range 0 to 255. Hence the hardware proposed reduces the time in distance computation by computing the distance between the centroids and 256 intensity level irrespective of the image size. The proposed hardware utilize less space compared to original K-means architecture. Also for an image size of 256X256, the clustering operation can be computed as fast as 203 frames per second which indicate that the proposed hardware is well suited for real time applications.

6. REFERENCES

[1] J. B. MacQueen, 1967 “Some methods for classification and analysis of multivariate observation”, In: Le Cam, L.M., Neyman, J. (Eds.), University of California.

[2] Yang, Bo, Xiao Fu, Nicholas D. Sidiropoulos, and Mingyi Hong. "Towards k-means-friendly spaces: Simultaneous deep learning and clustering." In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3861-3870. JMLR. Org, 2017.

[3] Windarto, Agus Perdana. "Implementation of Data Mining on Rice Imports by Major Country of Origin Using Algorithm Using K-Means Clustering Method." *International Journal of Artificial Intelligence Research* 1, no. 2 (2017).

[4] Jiang, Xiaoping, Chenghua Li, and Jing Sun. "A modified K-means clustering for mining of multimedia databases based on dimensionality reduction and similarity measures." *Cluster Computing* (2017): 1-8.

[5] G. d. S. Filho, A. C. Frery, C. C. de Araújo, H. Alice, J. Cerqueira,J. A. Loureiro, M. E. de Lima, M. d. G. S. Oliveira, and M. M. Horta, “Hyperspectral images clustering on reconfigurable hardware using the K-means

algorithm," in *Proc. Symp. Integr. Circuits Syst. Des.*, Sep.2003, pp. 99–104.

[6] Saegusa T, Maruyama T.: 'An FPGA implementation of real-time K-means clustering for color images'. Journal of Real-Time Image Processing. 2007 Dec 1; 2(4):309-18.

[7] Maruyama, Tsutomu. "Real-time k-means clustering for color images on reconfigurable hardware." In *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 2, pp. 816-819. IEEE, 2006.

[8] Chen, Tse-Wei, and Shao-Yi Chien.: "Bandwidth adaptive hardware architecture of K-means clustering for video analysis." *IEEE transactions on very large scale integration (VLSI) systems* 18, no. 6 (2009): 957-966.

[9] Chen, Tse-Wei, and Shao-Yi Chien. "Flexible hardware architecture of hierarchical K-means clustering for large cluster number." *IEEE transactions on very large scale integration (VLSI) systems* 19, no. 8 (2010): 1336-1345.

[10] T.-W. Chen, Y.-L. Chen and S.-Y. Chien, "Fast image segmentation based on K-Means clustering with histograms in HSV color space," in Proceedings of IEEE International Workshop on Multimedia Signal Processing, Oct. 2008, pp. 322–325.

[11] AC Frery, CCde Araujo, H Alice "Hyperspectral images clustering on reconfigurable hardware using the K-Means algorithm" in *Proc. IEEE Int. Symp. Circuits Syst.*, Sep 2003, pp. 94–104

[12] B. Maliatski and O. Yadid-Pecht, "Hardware-driven adaptive K-means clustering for real-time video imaging," *IEEE Trans. Circuits Syst.Video Technol.*, vol. 15, no. 1, pp. 164–166, Jan. 2005.

[13] Huang and D.-H. Liu, "Segmentation of color image using EM algorithm in HSV color space," in *Proceedings of IEEE International Conference on Information Acquisition*, Jul. 2007, pp. 316–319.

[14] S. J. Redmond, C. Heneghan, "A method for initializing the K-means clustering algorithm using kd-trees. Science direct", Pattern Recognition Letters 28 (2007) 965–973

[15] T.-W. Chen, C.-H. Sun, J.-Y. Bai, H.-R. Chen, and S.-Y. Chien, "Architectural analyses of K-Means silicon intellectual property for image segmentation," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2008, pp. 2578–2581.

[16] Anuradha, M. G., and L. Basavaraj. "Design and Implementation of High Speed VLSI Architecture of Online Clustering Algorithm for Image Analysis." *Data Engineering and Intelligent Computing*. Springer, Singapore, 2018. Pp. 197-206.