Unsupervised Image Segmentation Circuit Based on Fuzzy C-Means Clustering

Wen-Jyi Hwang, Zhe-Cheng Fan, Tsung-Mao Shen
Department of Computer Science and Information Engineering,
National Taiwan Normal University
Taipei, Taiwan
e-mails: whwang@ntnu.edu.tw; 699470137@csie.ntnu.edu.tw; 698470594@csie.ntnu.edu.tw

Abstract—**This paper presents a novel VLSI architecture for unsupervised image segmentation. The circuit is a hardware implementation of fuzzy *c*-means algorithm for the unsupervised clustering. The number of segments is determined by Xie-Beni index. An efficient pipeline circuit is proposed for the computation of the index. The circuit is used as a hardware accelerator of a softcore processor in a system-on-programmable chip for physical performance measurement. Experimental results reveal that the proposed architecture is an effective alternative for realtime segmentation with low error rate and area costs.**

Keywords-FPGA; Image Segmentation; Unsupervised Clustering; System-on-Chip.

I. INTRODUCTION

The goal of image segmentation is to cluster image pixels into multiple segments. The segmentation results can be used to identify regions of interest and objects in the scene for the subsequent image analysis or annotation. The fuzzy *c*-means algorithm (FCM) [1] is one of the most used techniques for image segmentation [2][3]. The effectiveness of FCM is due to the employment of fuzziness for the clustering of each image pixel.

Nevertheless, there are some drawbacks to employ the FCM algorithm. The first is its high computational complexity for membership coefficients computation and centroid updating. In addition, the size of membership matrix grows as the product of data set size and number of segments. As a result, the corresponding memory requirement may prevent the algorithm from being applied to large images. Finally, the number of segments should be pre-specified. Therefore, it is difficult to use FCM for the fully unsupervised realtime image segmentation.

A number of algorithms [4][5] have been proposed for accelerating the computational speed and/or reducing memory requirement of FCM. Most of these algorithms are implemented by software, and only moderate acceleration can be achieved. In [6][7], hardware implementations of FCM are proposed. However, the design in [6] is based on analog circuits. The clustering results therefore are difficult to be directly used for digital applications. Although the architecture shown in [7] adopts digital circuits, the architecture aims for applications with only two classes. The architecture may then not be useful for applications demanding the clustering of larger number of classes.

With the above observation, our earlier work [8] introduced a digital FCM architecture which can process more than two classes. Although the architecture is effective, its area cost is very high. The large hardware resource consumption arises from the employment of broadcasting scheme for membership coefficients and centroid computation at centroid level. As a result, the area cost grows with the number of segments. The FCM architecture may then only be used for clustering applications with small number of segments. Moreover, the architecture does not provide the function of determining the number of segments. The FCM architecture presented in [9] is able to reduce the area cost. However, the number of classes still needs to be pre-specified. There architectures are therefore not suited for the implementation of fully unsupervised realtime image segmentation.

The goal of this paper is to present a novel FCM architecture for fully unsupervised realtime image segmentation. In order to eliminate the large storage size for membership matrix, our implementation combines the usual iterative updating processes of membership matrix and cluster centroid into a single updating process. In our approach, the updating process is divided into three steps: pre-computation, membership coefficients updating, and centroid updating. The pre-computing step is used to compute and store information common to the updating of different membership coefficients. This step is beneficial for reducing the computational complexity for the updating of membership coefficients.

The membership updating step computes new membership coefficients based on a fixed set of centroids and the results of the pre-computation step. The weighted sum of data points and the sum of membership coefficients are also updated incrementally here for the subsequent centroid computation. This incremental updating scheme eliminates the requirement for storing the entire membership coefficients.

Following the updating process of membership matrix and cluster centroid, a cluster validation process is performed to find the optimal number of segments. The Xie-Beni index [10] is employed for this purpose because of its simplicity and effectiveness. Partial results of the updating process can be used for the computation of this index. In addition, an efficient pipeline architecture is proposed to further enhance the throughput of the computation.

For each class number, the updating process of membership matrix and cluster centroid, and the cluster validation process are performed sequentially. The resulting Xie-Beni index is stored, and compared with that associated with other class numbers. The class number with minimum index value is then selected as the class number for the image segmentation.

The proposed architecture has been implemented on field programmable gate array (FPGA) devices [11] so that it can operate in conjunction with a softcore CPU. Using the reconfigurable hardware, we are capable of constructing a system on programmable chip (SOPC) system for the physical performance measurement. Experimental results show that the proposed architecture has the advantages of high speed computation, low area cost and low error rate for image segmentation. In addition, because of its effectiveness, the proposed architecture can also be directly used for other clustering applications where the number of clusters is desired to be determined in an unsupervised manner such as spike sorting [12].

The remaining parts of this paper are organized as follows: Section 2 gives a brief review of the FCM algorithm. Section 3 describes the proposed FCM architecture. Experimental results are included in Section 4. Finally, the concluding remarks are given in Section 5.

II. PRELIMINARIES

This section gives a brief review of the FCM algorithm. Let $\boldsymbol{X} = \{x_1, \ldots, x_t\}$ be a data set to be clustered by the FCM algorithm into c classes, where t is the number of data points in the design set. Each class i, $1 \le i \le c$, is identified by its centroid v_i. For the image segmentation applications, $\boldsymbol{X}$ is an image to be segmented, x_k is a block in $\boldsymbol{X}$, t is the number of blocks in $\boldsymbol{X}$, and c is the class number. The goal of FCM is to minimize the following cost function:

$$J = \sum_{i=1}^{c}\sum_{k=1}^{t} u_{i,k}^{m} \left\| x_k - v_i \right\|^2 , \tag{1}$$

where $u_{i,k}^{m}$ is the membership of x_k in class i, and $m > 1$ indicates the degree of fuzziness. The cost function J is minimized by a two-step iteration in the FCM. In the first step, the centroids v_1, ..., v_c, are fixed, and the optimal membership matrix is computed by

$$u_{i,k} = \left(\sum_{j=1}^{c} \left(\left\| x_k - v_i \right\| / \left\| x_k - v_j \right\| \right)^{2/(m-1)} \right)^{-1} . \tag{2}$$

After the first step, the membership matrix is then fixed, and the new centroid of each class i is obtained by

$$v_i = \left(\sum_{k=1}^{t} u_{i,k}^{m} x_k \right) / \left(\sum_{k=1}^{t} u_{i,k}^{m} \right). \tag{3}$$

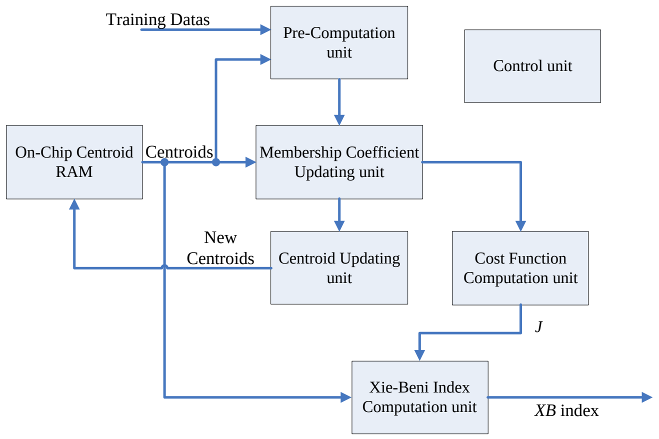


Figure 1. The proposed FCM architecture.

The FCM algorithm requires large number of floating point operations. Moreover, from (1) and (3), it follows that the membership matrix needs to be stored for the computation of cost function and centroids. As the size of the membership matrix grows with the product of t and c, the storage size required for the FCM may be impractically large when the data set size and/or the number of classes become high.

In the FCM, the number of classes c needs to be pre-specified. For the fully unsupervised image segmentation, the class number also needs to be determined. One way to find the optimal class number is to evaluate the clustering results for each c based on a cluster validation index. The class number producing the optimal index value is selected as the actual class number for image segmentation. A commonly used cluster validation index is the Xie-Beni index [10], which is defined as

$$XB(c) = \frac{J}{t\left(\min_{i \neq j} \| v_i - v_j \|^2 \right)} , \tag{4}$$

where J is the cost function of FCM defined in (1).

III. THE PROPOSED ARCHITECTURE

As shown in Figure 1, the proposed FCM architecture can be decomposed into six units: the pre-computation unit, the membership coefficients updating unit, the centroid updating unit, the cost function computation unit, the on-chip centroid RAM, and the control unit.

A. Precomputation Unit

The pre-computation unit is used for reducing the computational complexity of the membership coefficients calculation. Observe that (2) can be rewritten as

$$u_{i,j} = \left(\| x_k - v_i \| \right)^{-2/(m-1)} P_k^{-1} \tag{5}$$

where

$$P_k = \sum_{j=1}^{c} (1/\|x_k - v_j\|^2)^{1/(m-1)}. \quad (6)$$

Given x_k and centroids v_1, ..., v_c, membership coefficients $u_{1,k}$, ..., $u_{c,k}$ have the same P_k. Therefore, the complexity for computing membership coefficients can be reduced by calculating P_k in the pre-computation unit. For the sake of simplicity, we set $m = 2$ for our design.

Figure 2 shows the architecture of the pre-computation unit, where the x_k is obtained from the on-chip memory of the SOPC system, and v_i is obtained from the on-chip centroid RAM of the FCM architecture. As depicted in Figure 2, the circuit in its simplest form can be divided into two stages, which involve the squared distance computation, and inverse computation, respectively. The circuit can easily be separated into multistage pipeline for enhancing the throughput.

B. *Membership Coefficient Updating Unit*

Figure 3 depicts the architecture of the membership coefficients updating unit based on (5). It can be observed from Figure 3 that, given a training data x_k, the membership coefficients updating unit computes $u_{i,k}^2$ for i= 1,..., c, one at a time. Similar to the pre-computation unit, the x_k remains as the input until all the centroids v_i, i= 1,...,c, have been fetched from the on-chip centroid RAM for the computation of $u_{i,k}^2$. Based on (5) with $m = 2$, it follows that the circuit contains 3 multipliers and 1 divider. Similar to the precomputation unit architecture, the circuit can be separated into multistage pipeline for efficient computation.

C. *Centroid Computation Unit*

The centroid updating unit incrementally computes the centroid of each cluster. The major advantage for the incremental computation is that it is not necessary to store the entire membership coefficients matrix for the centroid computation. The centroid updating unit computes the incremental centroid when x_k and $u_{i,k}^2$ are received, and clusters will only be updated when the final centroid is generated after completing the computation of last training vector. Thus, no membership coefficients matrix is needed. Define the incremental centroid for the i-th cluster up to data point x_k as

$$v_i(k) = (\sum_{n=1}^{k} u_{i,n}^m x_n)/(\sum_{n=1}^{k} u_{i,n}^m). \quad (7)$$

When $k = t$, $v_i(k)$ is then identical to the actual centroid v_i given in (3).

Figure 4 shows the architecture of the centroid update unit, which contains a multiplier, an intermediate on-chip RAM and a divider. The unit has three inputs: centroid index i, training vector x_k and membership coefficient $u_{i,k}^2$. As shown in Figure 4, both $u_{i,k}^2 x_k$ and $u_{i,k}^2$ are used as the inputs to the intermediate on-chip RAM for computing $v_i(k)$.

D. *Cost Function Computation Unit*

Similar to the centroid updating unit, the cost function unit incrementally computes the cost function J. Define the incremental cost function $J(k)$ up to data point x_k as

$$J(i, k) = \sum_{z=1}^{k} \sum_{j=1}^{i} u_{j,z}^2 \|x_z - v_j\|^2. \quad (8)$$

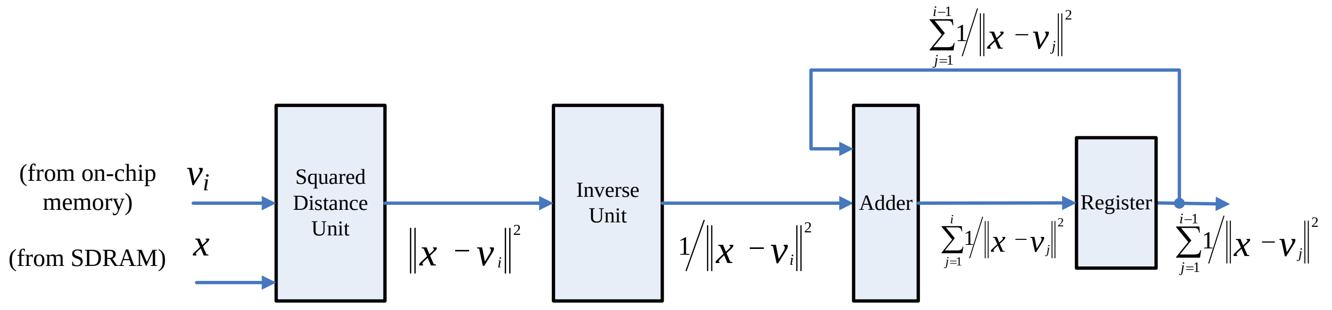


Figure 2. The architecture of precomputation unit.

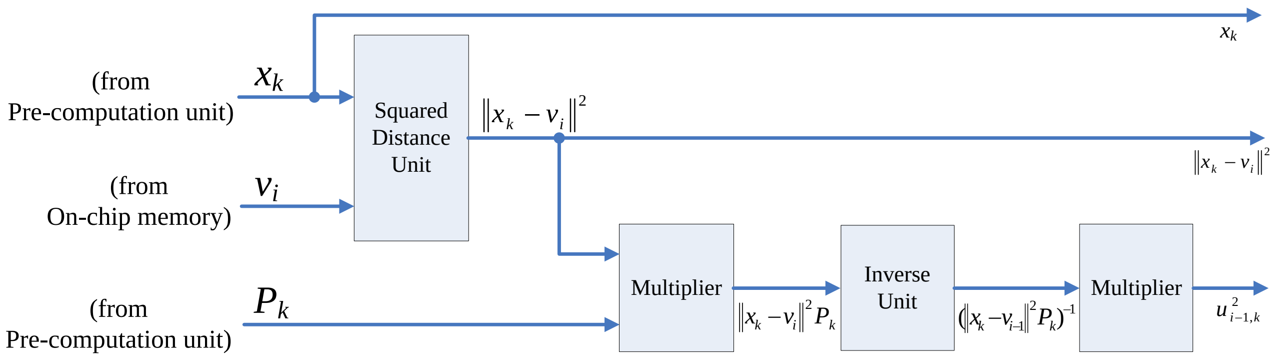


Figure 3. The architecture of membership coefficient updating unit

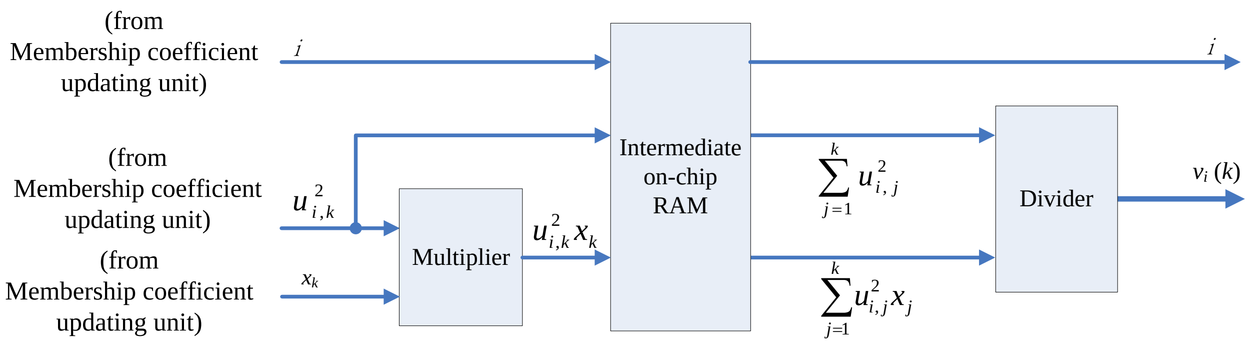

(from Membership coefficient updating unit)
i
(from Membership coefficient updating unit)
$u^2_{i,k}$
(from Membership coefficient updating unit)
x_k
Multiplier
$u^2_{i,k} x_k$
Intermediate on-chip RAM
i
$\sum_{j=1}^{k} u^2_{i,j}$
$\sum_{j=1}^{k} u^2_{i,j} x_j$
Divider
$v_i\ (k)$

Figure 4. The architecture of centroid computation unit

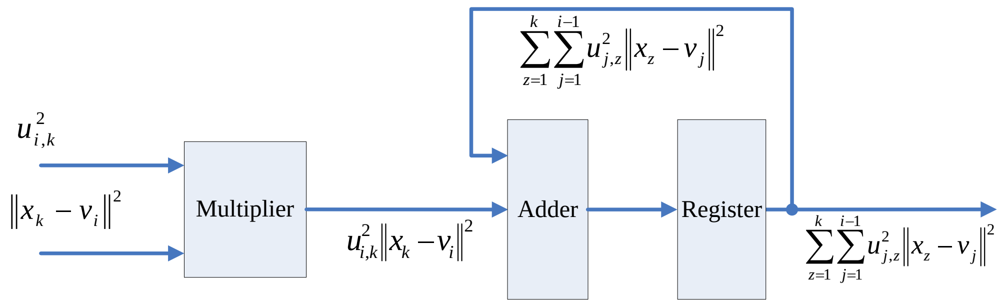

$u^2_{i,k}$
$\|x_k - v_i\|^2$
Multiplier
$u^2_{i,k}\|x_k - v_i\|^2$
Adder
Register
$\sum_{z=1}^{k}\sum_{j=1}^{i-1} u^2_{j,z}\|x_z - v_j\|^2$
$\sum_{z=1}^{k}\sum_{j=1}^{i-1} u^2_{j,z}\|x_z - v_j\|^2$

Figure 5. The architecture of cost function computation unit

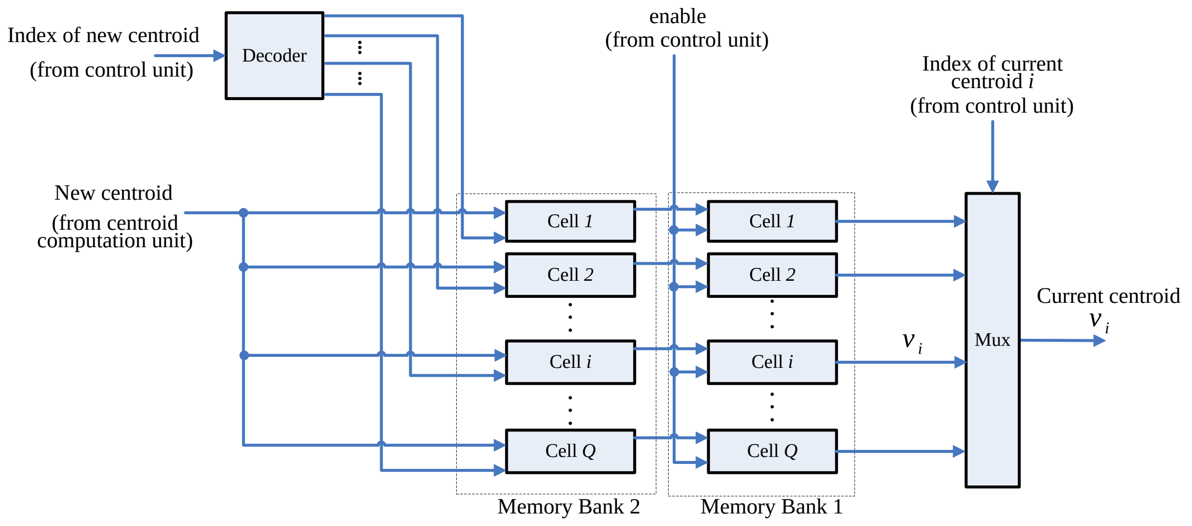


Figure 6. The architecture of on-chip centroid RAM

As shown in Figure 5, the cost function computation circuit receives $u_{i,k}^2$ and $\|x_k - v_i\|^2$ from the membership coefficients updating unit. The product $u_{i,k}^2\|x_k - v_i\|^2$ is then accumulated for computing $J(i, k)$ in eq. (7).

When $i = c$ and $k = t$, $J(i, k)$ then is identical to the actual cost function J given (1). Therefore, the output of the circuit becomes J as the cost function computations for all the training vectors are completed.

E. On-Chip Centroid RAM

This unit is used for storing the centroids for FCM clustering. An shown in Figure 6, there are two memory banks (Memory Bank 1 and Memory Bank 2) in the on-chip centroid RAM. The Memory Bank 1 stores the current centroids v_1, ..., v_c. The Memory Bank 2 contains the new v_1, ..., v_c obtained from the centroid updating unit. Only the centroids stored in the Memory Bank 1 are delivered to the pre-computation unit and membership updating unit for the membership coefficients computation. The updated centroids obtained from the centroid updating unit are stored in the Memory Bank 2. Note that, the centroids in the Memory Bank 2 will not replace the centroids in the Memory Bank 1 until all the input training data points x_k, $k = 1, ..., t$, are processed.

It can also be observed from Figure 6 that there are Q cells in each memory bank, where Q is the upper limit of the number of centroids c. Therefore, the proposed FCM circuit is able to conduct image segmentation with number of classes c less than or equal to Q.

F. Xie-Beni Index Computation unit

The goal of Xie-Beni Index computation unit is to compute $XB(c)$ given in (4). The numerator of $XB(c)$ is actually the cost function. Hence, we can directly use the output of the cost function unit as the numerator of $XB(c)$.

The denominator contains $\min_{i,k} \|v_i\text{-}v_k\|$. The corresponding circuit should be implemented in the cluster validity index computation unit. Although the direct implementation of $\min_{i,k} \|v_i\text{-}v_k\|$ is possible, the time and area complexity would be $O(c^2)$. Therefore, the complexities would be very high when c becomes large. The proposed circuit is able to reduce the overhead. Figure 7 shows the architecture of Xie-Beni index computation unit, which contains the minimum computation unit, a multiplier, and a divider. The minimum computation unit contains an efficient pipeline for the computation of $\min_{i,k}\|v_i\text{-}v_k\|$, as depicted in Figure 8. The circuit can be viewed as a c–stage pipeline, where each stage contains one processing module (PM). The centroids are delivered to the pipeline from on-chip centroid memory one at a time. Each centroid will traverse through the pipeline. As shown in the figure, the latest input entering the pipeline will be broadcasted to all the PMs. Let

$$D_{min}(v_p) = \min_{k,k\neq p} \| v_p - v_k \|^2 . \tag{9}$$

Suppose now the centroid v_p arrives at PM i, and the centroid v_q is the newest centroid entering the pipeline. In the PM i, the distance between v_p and v_q will be computed, and will be compared with the current $D_{min}(v_p)$. If $\|v_p\text{-}v_q\|^2 <$ current $D_{min}(v_p)$, then $\|v_p\text{-}v_q\|^2$ will be the new current $D_{min}(v_p)$. As v_p reaches stage c of the pipeline, the current $D_{min}(v_p)$ becomes the actual $D_{min}(v_p)$. When all the centroids have reached the stage c, the actual $\min_{i,k} \|v_i\text{-}v_k\|$ can be computed by

$$\min_{i\neq j} \| v_i - v_j \|^2 = \min_p D_{min}(v_p). \tag{10}$$

The time and area complexities of the proposed pipeline are only $O(c)$. The proposed architecture is therefore effective for Xie-Beni index computation. Finally, we note that, because it is necessary to compute the $XB(c)$ for various c values, the pipeline actually will be implemented in Q stages, where Q is the upper bound of the c value.

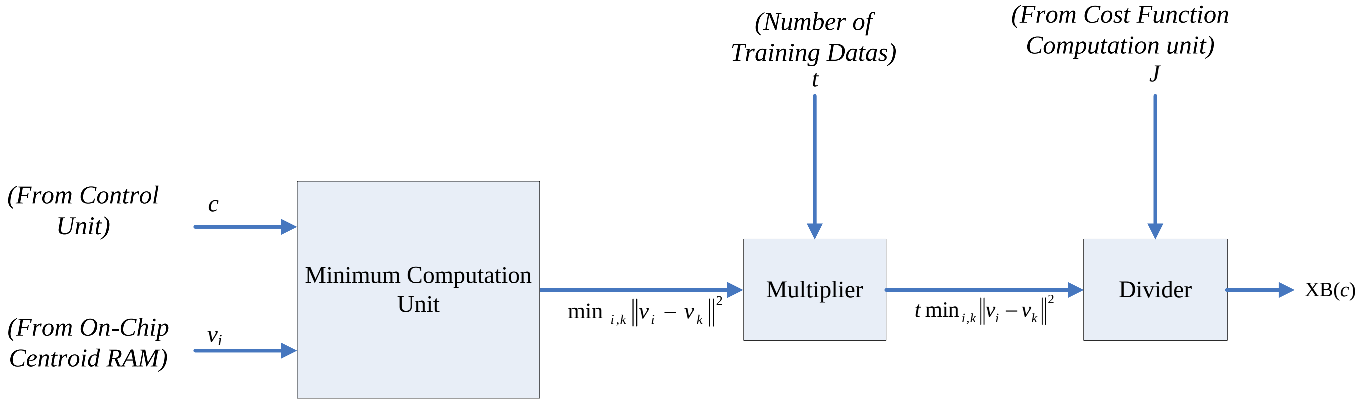

(Number of Training Datas)
t
(From Cost Function Computation unit)
J
(From Control Unit)
c
(From On-Chip Centroid RAM)
v_i
Minimum Computation Unit
$\min_{i,k}\|v_i - v_k\|^2$
Multiplier
$t\min_{i,k}\|v_i - v_k\|^2$
Divider
XB(c)

Figure 7. The architecture of Xie-Beni index computation unit.

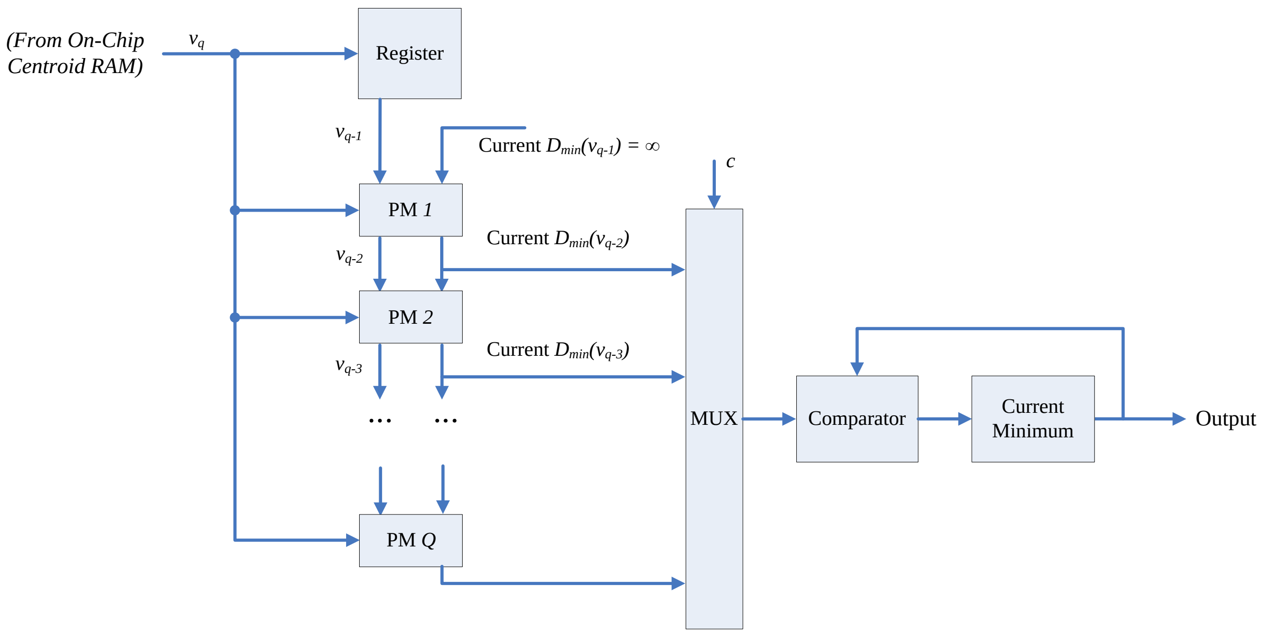
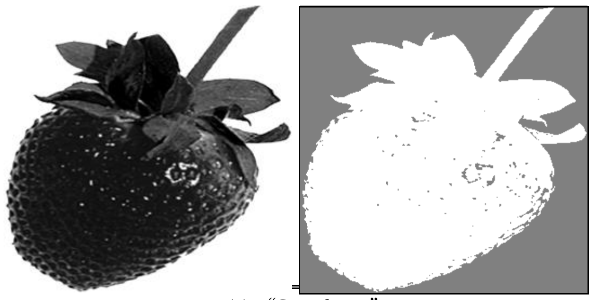

(From On-Chip Centroid RAM)
v_q
Register
v_{q-1}
Current $D_{min}(v_{q-1}) = \infty$
c
PM *1*
v_{q-2}
Current $D_{min}(v_{q-2})$
PM *2*
v_{q-3}
Current $D_{min}(v_{q-3})$
…
PM *Q*
MUX
Comparator
Current Minimum
Output

Figure 8. Architecture of minimum computation unit


(a) "Strawberry"

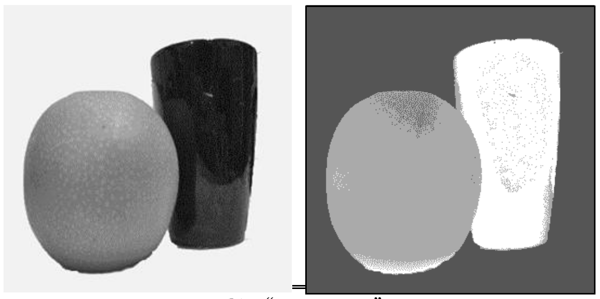
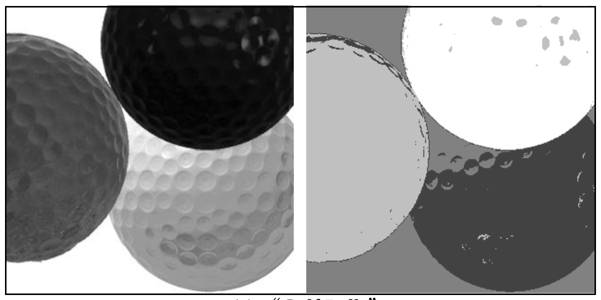

(b) "Pear & Cup"

Figure 9. The original images and their segmentation results produced by the proposed FCM architecture: (a) "Strawberry," (b) "Peer & Cup."


(a) "Gulf Balls"

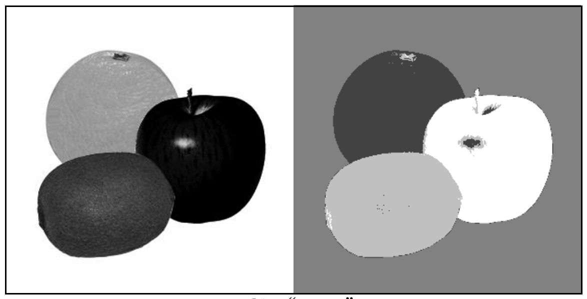

(b) "Fruits"

Figure 10. The original images and their segmentation results produced by the proposed FCM architecture: (a) "Gulf Balls," (b) "Fruits."

IV. EXPERIMENTAL RESULTS

This section presents some physical performance measurements of the proposed FPGA implementation. The design platform of our system is Altera Quartus II 8.0 with SOPC Builder and NIOS II IDE.

Figures 9 and 10 shows the segmentation results of the proposed FCM architecture with Q=10. Therefore, the circuit is able to conduct fully unsupervised segmentation for images with number of classes c less or equal to 10.

Table I. The estimated and actual number of classes, and the segmentation success rate of the proposed FCM architecture for the images shown in Figures 9 and 10

Images	Strawberry	Peer & Cup	Gulf Balls	Fruits
Est. Class Number $\hat{c}$	2	3	4	4
Actual Class Number c	2	3	4	4
Segmentation Success Rate	98.97%	97.13%	94.77%	98.87%

Table II. The CPU time of various FCM implementations

Q	Proposed Architecture	Basic Software FCM	Fast Software FCM [5]
2	15.47 ms	256.20 ms	62.40 ms
3	30.97 ms	709.45 ms	151.65 ms
4	50.05 ms	1404.70 ms	272.65 ms
5	69.11 ms	2404.70 ms	428.85 ms
6	88.20 ms	3720.30 ms	613.05 ms
7	107.28 ms	5410.90 ms	825.45 ms
8	126.36 ms	7535.90 ms	1067.78 ms
9	145.46 ms	10149.15 ms	1345.03 ms
10	164.55 ms	13389.75 ms	1648.23 ms

All the images have the same dimension 320 × 320. The images are separated into 2 × 2 blocks for FCM training and segmentation. Table 1 shows the estimated and actual number of classes, and the segmentation success rate of these images. The segmentation success rate of an image is defined as the number of pixels which are misclassified divided by the total number of pixels of the image. From Figures 9 and 10, and Table 1, it can be observed that the proposed architecture is able to correctly identify the number of classes with high classification success rate.

The speed of various FCM implementations is revealed in Table 2. The target FPGA device is Altera Stratix III EP3SL150F1152C2N [13]. The speed of the proposed architecture is the CPU time of the softcore NIOS processor [14] using the proposed architecture as the hardware accelerator. The clock rate of the NIOS processor is 75 MHz. The software implementations are running on 2.8 GHz Intel Pentium D processor. Two software implementations are considered: the basic FCM implementation, and the fast FCM implementation [5].

Figure 10 shows the speedup of the proposed architecture over the fast FCM [5]. It can be observed from Table 2 and Figure 11 that the proposed architecture has significantly lower computation time as compared with its software counterparts. Although the NIOS processor is running at a lower clock rate as compared with Intel CPU (i.e., 75 MHz versus 2.8 GHz), it still has higher computational speed because of the efficiency of the proposed architecture for the membership matrix and centroid computation.

The hardware utilization of the proposed architecture for various *Q* values is shown in Table 3 for Altera Stratix III EP3SL150F1152C2N. It can be observed from the table that the consumption of ALMs and DSP block grow linearly with *Q*. Nevertheless, only a small fraction of hardware resources are consumed. In particular, when *Q*=10, only 20 %, 27 % and 19% of the ALM, block memory bits, and DSP blocks are consumed by the proposed architecture.

Finally, Table 4 compares the hardware utilization of the proposed architecture with that of the architecture in [8] with block size 2×2. The target device is Altera Cyclone III EP3C120. The logic elements (LEs) are the hardware resources considered in the table.

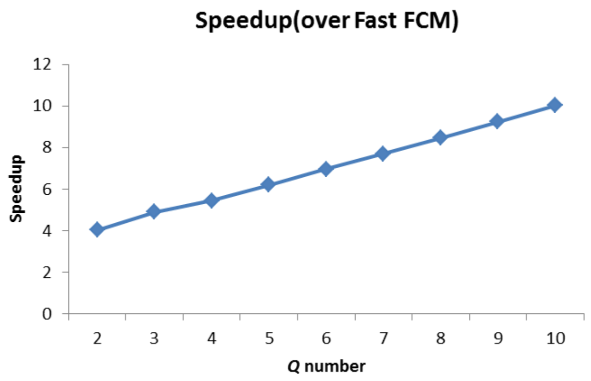


Figure 11. The speedup of the proposed architecture over the fast FCM in [5].

From Table 4, we can see that the proposed architecture has significantly lower utilization of LEs as compared with the architecture in [8]. In fact, the proposed architecture is able to operate up to *Q*=64 with the consumption of only 40% of LEs of the target FPGA. By contrast, the architecture in [8] consumes almost all the LEs when *Q* reaches 32. All these facts demonstrate the effectiveness of the proposed architecture.

Table III. The hardware utilization of the proposed architecture.

Q	ALMs	Block Memory Bits	DSP Block Elements
2	10738/56800 (18%)	1535264/5630976 (27%)	40/384 (10%)
3	10814/56800 (19%)	1535840/5630976 (27%)	44/384 (11%)
4	10893/56800 (19%)	1535904/5630976 (27%)	48/384 (13%)
5	11056/56800 (19%)	1536992/5630976 (27%)	52/384 (14%)
6	11199/56800 (19%)	1537056/5630976 (27%)	56/384 (15%)
7	11308/56800 (19%)	1537120/5630976 (27%)	60/384 (16%)
8	11405/56800 (20%)	1537184/5630976 (27%)	64/384 (17%)
9	11612/56800 (20%)	1539296/5630976 (27%)	68/384 (18%)
10	11793/56800 (20%)	1539360/5630976 (27%)	72/384 (19%)

Table IV. The LE utilization of various architectures.

Q	Proposed Architecture	Architecture in [8]
4	16553/119088 (14%)	21084/119088 (18%)
8	18504/119088 (16%)	35423/119088 (30%)
16	22568/119088 (19%)	59868/119088 (50%)
32	30827/119088 (26%)	114117/119088 (97%)
64	47412/119088 (40%)	N/A

V. CONCLUDING REMARKS

Experimental results revealed that the proposed architecture is able to correctly estimate the number of classes of an image with segmentation success rate above 94%. For the cases where the upper bound of the number of classes is 10, the proposed architecture consumes less than 30% of the ALMs, block memory bits, and DSP blocks of the Stratix III FPGA device. It also attains speedup of 10 over its software counterpart running on the Intel general purpose CPU. The proposed architecture, therefore, is effective for unsupervised image segmentation with low area costs and high computation speed.

REFERENCES

[1] J.C. Bezdek, *Fuzzy Mathematics in Pattern Classification*, Cornell University: Ithaca, NY, USA, 1973.

[2] S.C. Chen and D.Q. Zhang, “Robust image segmentation using FCM with spatial constraints based on new kernel-induced distance measure,” IEEE Trans. Syst. Man Cybern. B, 2004, pp. 1907-1916.

[3] K.S. Chuang, H.L. Tzeng, S. Chen, J. Wu, and T.J. Chen, “ Fuzzy c-means clustering with spatial information for image segmentation,” Comput. Med. Imaging Graphics, 2006, pp. 9-15.

[4] S. Eschrich, J. Ke, L.O. Hall, and D.B. Goldgof, “Fast Accurate Fuzzy Clustering Through Data Reduction,” IEEE Transaction on. Fuzzy Systems, 2003, pp. 262-270.

[5] J. F. Kolen and T. Hutcheson, “Reducing the Time Complexity of the Fuzzy C-Means Algorithm,” IEEE Trans. Fuzzy Systems, pp. 263-267, Vol. 10, 2002.

[6] J. Garcia-Lamont, L.M. Flores-Nava, F. Gomez-Castaneda, and J.A. Moreno-Cadenas, “CMOS Analog Circuit for Fuzzy C-Means Clustering,” IEEE Proceedings 5th Biannual World Automation Congress, 2002.

[7] J. Lazaro, J. Arias, J.L. Martin, C. Cuadrado, and A. Astarloa, “Implementation of a Modified Fuzzy C-Means Clustering Algorithm for Realtime Applications,” Microprocessors and Microsystems, 2005, pp. 375-380.

[8] H.Y. Li, C.T. Yang, and W.J. Hwang, “Efficient VLSI Architecture for Fuzzy C-Means Clustering in Reconfigurable Hardware,” Proc. IEEE International Conference on Frontier of Computer Science and Technology, 2009, pp. 168-174.

[9] H.Y. Li, W.J. Hwang, and C.Y. Chang, “Efficient Fuzzy C-Means Architecture for Image Segmentation”, Sensors, 2011, pp.6697-6718.

[10] X.L. Xie and G. Beni. “A Validity measure for Fuzzy Clustering”, IEEE Transactions on Pattern Analysis andmachine Intelligence, 1991.

[11] S. Hauck and A. DeHon, *Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation*, Morgan Kaufmann, USA, 2008.

[12] M.S. Lewicki,“A review of methods for spike sorting: the detection and classification of neural action potentials,” Network Computer Neural System, 1998, pp. R53-R78.

[13] Altera Corporation, *Stratix III Device Handbook*, 2011, http://www.altera.com/literature/lit-stx3.jsp (accessed on 6 August, 2012).

[14] Altera Corporation, *NIOS II Processor Reference Handbook*, 2011, http://www.altera.com/literature/lit-nio2.jsp (accessed on 6 August, 2012).