A Fast Massively Parallel Fuzzy C-Means Algorithm for Brain MRI Segmentation

Mohamed Youssfi [a,✉] , Omar Bouattane [a] , Mohammed Ouadi Bensalah [b], Bouchaib Cherradi [c,a]

[a] Laboratory SSDIA, E.N.S.E.T, University Hassan II of Casablanca, Morocco
[b] FSR, University Mohammed V, Rabat, Morocco
[c] CRMEF El jadida, Morocco
Tel : +212 661326837 E-mail: med@youssfi.net

Abstract

In this paper, we propose a fast parallel algorithm for data classification, and its application for Magnetic Resonance Images (MRI) segmentation. The presented classification method is based on a parallel fine grained fuzzy C-means algorithm. It is implemented on a polymorphic SIMD machine to sort out the different components of a brain image. The use of the massively parallel architecture in the classification domain and particularly for the fuzzy classification is introduced to improve the complexities of the corresponding algorithms. The proposed algorithm is assigned to be implemented on a massively parallel machine, which is the Reconfigurable Mesh Computer (RMC). The brain image of size (m x n) to be processed must be stored on the RMC of the same size, one pixel per Processing Element (PE). Some interesting results are obtained in terms of accuracy and efficiency analysis of the proposed method, thanks to the reconfiguration ability of the used computational model.

Keywords: Parallel Programming, Image Segmentation, Brain MRI image, Parallel Classification, Fuzzy C-means, Massively Parallel Algorithm.

1. Introduction

Magnetic Resonance (MR) imaging has been widely used in brain exploration, due to its excellent soft tissue contrast, non-invasive behavior, high spatial resolution and easy slice selection at any orientation. However, accurate and fast tools for cerebral MR images processing are of great interest for many brain manipulations such as analysis, interpretation, diagnostics, and examination of the progression of brain disorders such as Alzheimer's disease, multiple sclerosis or schizophrenia, and neurosurgical operation planning [1].

MRI segmentation can be considered as an Image processing problem or a pattern recognition one. In both cases, the problem is to classify a set of elements into a set of classes according to the shared characteristics inside the clusters. In the MRI segmentation domain, the vector pattern X that will be considered in the FCM algorithm corresponds to the gray level of the studied pixel in each MRI slice.

In the medical imaging field clustering is usually used for pattern recognition (Brain retrieval, Tumor segmentation etc…). The corresponding clustering algorithms require often a huge volume of data computation. To achieve clustering result rapidly, several computational models have been proposed as the high performance tools to improve performance and efficiency of the proposed method.

Actually, the massively parallel architectures are known as the high performance computational models. They have demonstrated their effectiveness in terms of supporting the most complex parallel algorithms such as Fuzzy C-Mean algorithms [1].

In [2], the authors have discussed the need of parallel methods to speed up clustering algorithms. They have implemented their parallel FCM on a parallel architecture named "Red hat based cluster". This parallel system, possesses eight nodes supervised by a blade node using c programming language an message passing instruction (MPI) model to implement the parallel proposed algorithm. They have notified an interesting speed up result when the number of nodes increases.

In [3], authors have proposed a parallel Fuzzy C-Mean clustering for large data set (PFCM). This later is assigned o be implemented on a parallel computer of Single Program Multiple Data (SPMD) model using MPI tool. They compare the obtained scalability and parallel capability to an existing parallel C-Mean algorithm.

Due to the huge amount of data to be processed in the classification domain, several parallel implementations have appeared, both in distributed and shared memory hardware as well as in grid environment [4]. All the proposed algorithms have been investigated as : Decision tree induction [5] , Fuzzy rule based classifiers [6] [7] , neural networks [8] [9] , association rules mining [10] [11] and clustering [2] [12].

Since the number of proposed computational models and methods is large, we notify that several associated algorithms have been proposed such as: c-means [14], fuzzy cmeans (FCM) [15], adaptive c-means [16], modified fuzzy cmeans [17] using illumination patterns and fuzzy c-means combined with neutrosophic set [18].

Image segmentation is a splitting process of images into a set of regions, classes or homogeneous sub-sets according to some criteria. Usually, Gray levels, texture or shapes constitute the well used segmenting criteria. Frequently the criterion choice is based on the kind of images and the target goals after processing. Image segmentation can be considered as an Image processing problem or a pattern recognition one. In the case of Image processing, we distinguish two essential approaches that are: region approach and contour approach. In the first approach, one looks for homogeneous regions using some basic techniques such as thresholding, region growing, morphological mathematics and others [19]. Thresholding technique discriminates pixels using their gray levels. It supposes implicitly that intensity structures are sufficiently discriminative, and guarantee good separation [20]. In [21] authors, propose a multi-modal histogram thresholding method for gray-level images. As mentioned above, segmentation represents a very large problem in the image processing domain; it requires several algorithmic techniques and different computational models, which can be sequential or parallel using processing elements, cellular automata, neural networks or other advanced tools.

In this paper, we propose a massively parallel algorithm for fuzzy classification (fuzzy c-means) and its application to the MRI cerebral images. The proposed algorithm is assigned to be implemented on a SIMD structure which is an (n x n) massively parallel Reconfigurable Mesh Computer (RMC). It is a fine grained version instead of the most proposed algorithms in the literature. These later are based on the “Message Passing Interface” (MPI) tool in the coarse grained parallel structure having a reduced number of nodes (e.g. at most 16 nodes). Our fine grained parallel algorithm requires computational model of the same size (n x n) as the image, where each pixel (i, j) is associated to it corresponding processing element PE(i, j). In this context, the proposed PFCM algorithm is assigned to an upper bound theoretic machine to see at first how to reach the real time of the fuzzy clustering algorithm and secondly how to implement this algorithm in a real machine in order to discuss the dynamic evolution of the class centers according to the data input image and to appreciate its usefulness in the medical imaging domain.

To validate the proposed method, we use an emulation platform of the RMC architecture [23], [24] where the developed parallel program is performed. Note that with this platform we can create several massively parallel virtual machines of any topology by associating the performances of all the available hardware resources (Desktop, GPU, Supercomputer, Cellular automata etc.). The polymorphic aspect of this platform allows researchers to build their own virtual parallel architectures for their specific problems even if theses architectures are not available. By this concept, we show the power and the high performance of our vision for the future to not restrict researchers only to the available technologies.

This paper is organized as follows: Section 2 presents a summary the computational model used to implement our parallel algorithm. The parallel segmentation fuzzy c-means algorithm and its implementation program code are described in more details in section 3, and the obtained segmentation results are presented in section 4. The complexity analysis of the parallel fuzzy c-means program and its improvement are discussed in the next section 5. Finally, the last section gives some concluding remarks on this work.

2. Parallel Computational Model

2.1. Presentation

The computational model that will support the proposed Parallel FCM is the Reconfigurable Mesh Computer (RMC) of same size n x n as the input MRI image. It is a massively parallel machine having n^2 Processing elements (PEs) arranged on a 2-D matrix as shown in figure 1. It is a Single Instruction Multiple Data (SIMD) structure, in which each PE(i, j) is localized in row i and column j and has an identifier defined by ID= n x i + j. Each PE of the mesh is connected to its four neighbors (if they exist) by communication channels. It has a finite number of registers of size ($\log_2$ n) bits. The PEs can carry out arithmetic and logical operations. They can also carry out reconfiguration operations to exchange data over the mesh. Also the PEs can use another way to exchange data using a shared memory. All the PEs are managed by a host manager that is designed to load parallel program and global data to distribute them over the matrix of PEs for any execution stage of the algorithm.

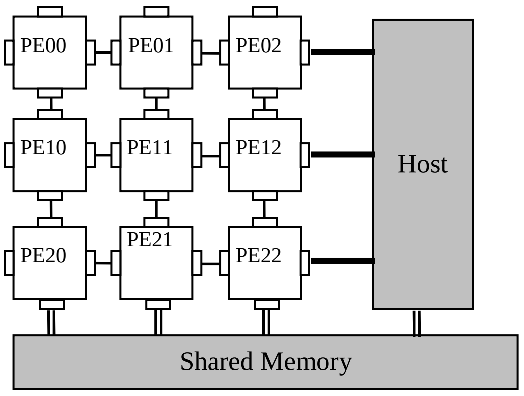


Figure 1. A Reconfigurable Mesh Computer of size 3 x 3.

2.1.1. Processing Element (PE) Model.

Like any processor, each processing element (PE) of the RMC can execute a set of instructions relating to the arithmetic and logical operations. The concerned operands can be the local data of a PE or the data arising on its communication channels after data exchange operation between the PEs. Figure 2 shows a simplified model of the processing element used in the considered bidirectional RMC (2D RMC).

The PE can also carry out the bridge configurations in order to establish connections between two or more communication channels. When the PE of the RMC is in a Simple Bridge (SB) state, it establishes connections between two of its communication channels. This PE can connect itself to each bit of its channels, either in transmitting mode, or in receiving mode, as it can be isolated from some of its bits (i.e. neither transmitter, nor receiver). Various SB configurations are described by the following formats:

{E, W, **S-N** },{E-S, W, N }, {N-W, S, E }, {E–N, S, W } and {W-S, E, N } where E, W, N and S indicate the East, West, North and South Ports of a given PE respectively.

For example, the format: {**W-S**, E, N } is a simple bridge configuration where the west and south ports are linked by communication channel. The East and north ports remain free.

A PE is in a Double Bridge (DB) state when it carries out a configuration to create two independent buses. Thus, the possible configurations of the DB state are: {EW, NS}, {ES, NW} and {EN, SW}.

A PE is in CB State when it connects all its active communication channels in only one; each bit with its correspondent. This operation is generally used when we want to transmit information over the mesh to a set of PEs at the same time. The CB state is defined by the configuration: {NESW}

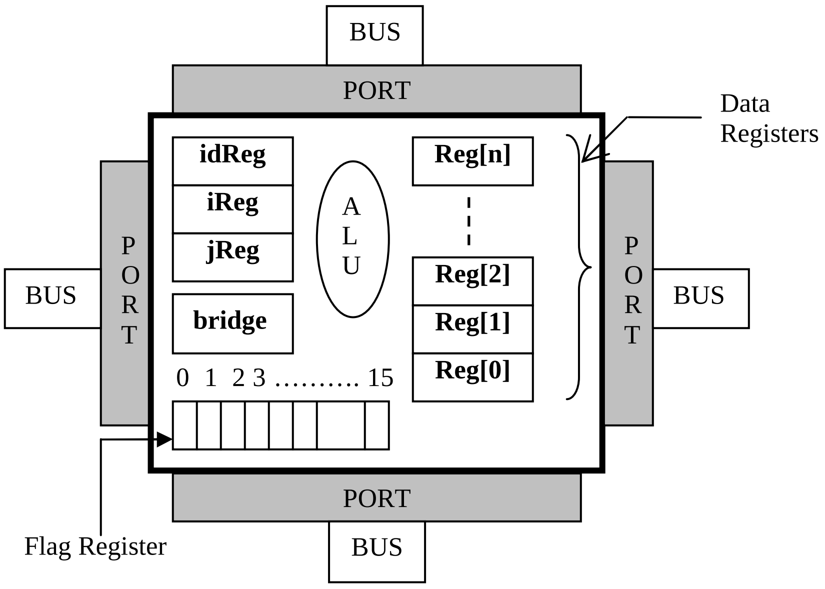


Figure 2. Different components of a processing element model of the 2D RMC model.

Notice that in the proposed PE model, the list of internal elements is not exhaustive. During the programming phase, we will try to deploy a minimal number of components resources (registers, flags, ports, etc.) to optimize the used RMC and to accomplish efficiently the FCM algorithm execution.

3. Parallel Fuzzy Segmentation Algorithm

3.1. Standard fuzzy c-means algorithm

The fuzzy *c*-means (FCM) clustering algorithm was first introduced by [22] and later was extended by [15]. Fuzzy C-means (FCM) is a clustering technique that employs fuzzy partitioning such that a data point can belong to all classes with different membership grades between 0 and 1.

The aim of FCM is to find the final values of the C cluster centers (centroids) in the data set $X=\{x_1, x_2, ..., x_N\}$ that minimize the following dissimilarity function:

$$J(U, V_1, V_2, \dots, V_C) = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}{}^m d^2(V_i, x_j) \qquad (1)$$

With the constraints:

$$u_{ij} \in [0,1], \forall i, j \qquad (2a)$$

$$\sum_{i=1}^{C} u_{ij} = 1, \forall j = 1, \dots, N \qquad (2b)$$

$$0 < \sum_{j=1}^{N} u_{ij} < N, \forall i = 1, \dots, C \qquad (2c)$$

Where:

u_{ij}: Membership of data xj in the cluster V_i;

V_i : Centroid of cluster i;

$d_{(Vi,xj)}$: Euclidian distance between ith centroid (V_i) and jth data point x_j;

m ϵ [1,∞] : fuzzy weighting exponent (generally equals 2).

N: Number of data.

C: Number of clusters $2 \leq C < N$.

To reach a minimum of dissimilarity function there are two conditions.

$$V_i = \frac{\sum_{j=1}^{N} u_{ij}{}^m x_j}{\sum_{j=1}^{N} u_{ij}{}^m} \qquad (3)$$

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d_{ij}}{d_{kj}} \right)^{2/(m-1)}} \qquad (4)$$

In order to implement the FCM algorithm even on serial or parallel computational model, it is necessary to build it around the following stages:

Stage 1: - Randomly initialize the membership matrix (U) according to the constraints of Equations 2a, 2b and 2c.

- initialize the fuzzification parameter *m* $(1 < m < \infty)$,
- Choose the number of clusters C,
- Initialize the initial values of cluster centers $V_i^{(0)}$ *(i=1 to c)* and threshold $S_{th} > 0$.

For each iteration *k*:

Stage 2: Calculate centroids $V_i^{(k)}$ using Equation (3).

Stage 3: Compute the objective function J by equation (1).

Stage 4: compare the obtained objective function J_k to J_{k-1} and exit the loop (i.e. go to stage 6) if the absolute value of the difference between the successive objectives functions is lower than S_{th}.

Stage 5: Compute a new membership matrix U using Equation (4) and Go to Stage 2.

Stage 6: Stop.

3.2. Parallel fuzzy c-means algorithm

The parallel algorithm, described in this section, is implemented in an RMC emulating framework [23] [24] using its XML based parallel programming language. The program code presented, in each stage of this parallel algorithm, is performed on the MRI cerebral image as input data. Before its execution, we define in the initialization phase the number of classes by c = 3. This means that the classes looked for in the image are the white matter, the gray matter and the cerebrospinal fluid. The background of the image is not considered by the algorithm.

3.2.1. Initialization procedure

– Loading data image input and initialization phase: Each PE(i,j) of the mesh will upload in its register 0 a pixel gray level of the image.

```
<loadImage file="brain.jpg" reg="0" codage="8"/>
```

– Loading initial parameters in the host registers :

```
<host>
  <loadValues regs="0,1,2,3,4,5" value="140,149,150,0,0,0.1"/>
  <!-- this instruction means that :
    reg[0]contains the initial value of C1 class center
     reg[1]contains the initial value of C2 class center
    reg[2]contains the initial value of C3 class center
    reg[3]contains the initial value of Jn-1
    reg[4]contains the initial value of Jn-1
    reg[5]contains the initial value of Sth
 -->
</host>
```

3.2.2. Class determination procedure

This procedure consists of six essential stages which are:

1- Data Broadcasting
2- Distance computation
3- Local objective function computation and new class center determination
4- Class centers and global objective function computation.
5- Loop stop test
6- Membership decision.

These various stages are included in a loop as follows:

<Beginning of iteration (k) >

3.2.2.1. Data broadcasting

```
For (c = 1 to C)
      {
         ▪ All the PEs of the RMC configure themselves in Crossed Bridge (CB) state.
         ▪ The representative PE of the class $V_c$ broadcasts its value $V_{c,k}$ through the mesh.
         ▪ All the PEs of the RMC store in their $Rx_m$ the value $V_{m,n}$ received through the mesh.
         ▪ All the PEs of the RMC stores the fuzzyfication parameter $m$ .
      }
```

It should be noted that the representative PE of a class is the PE having the smallest identifier ID in its class. The search for this smallest identifier calls a procedure of the Min-search in a group of PEs as in [13].

The corresponding parallel code of the data broadcasting procedure is :

```
<doWhile test="reg[30]>reg[5]" target="host">
  <!-- All PES except PEs representing background color  -->
    <for-eachPE test="reg[0]!=0">
         <mark type="true"/>
  <!-- All the PE’s load the three class centers from the host -->
      <loadValue regs="8,9,10" value="host.reg[0], host.reg[1],
      host.reg[2]"/>
  <!-- Loading the fuzzyfication parameter into reg[40] -->
      <loadValue reg="40" value="2"/>
```

3.2.2.2. Distance computation

At each iteration k, each PE computes the distances d(Ng, $V_{m,k}$) between its gray level Ng and the values of the C class centers. These values are stored in its C registers ($V_{1,k}$, $V_{2,k}$,… , $V_{C,k}$).

```
<!—-  Each PE computes the three distances separating it from the
three class centers. d1, d2 and d3 are stored in reg[11], reg[12]
and reg[13] respectively
-->
     <doOperation expression="reg[11]=Math.abs(reg[0]-reg[8])"/>
     <doOperation expression="reg[12]=Math.abs(reg[0]-reg[9])"/>
     <doOperation expression="reg[13]=Math.abs(reg[0]-reg[10])"/>
```

3.2.2.3. Local objective function computation:

a) Computing membership of data Xj to the cluster V_i

```
<!—
Each PE computes its membership degrees U1, U2 and U3 to the clus-
ters centred at C1, C2 and C3. The values of C1,C2 and C3 are re-
spectively stored in the registers reg[41], reg[42] and reg[43].
-->
<if test="(reg[11]!=0)and(reg[12]!=0)and(reg[13]!=0)">
<!—-  compute U1 -->
<doOperation expression= "reg[41]=
1/(Math.pow(reg[11]/reg[11],2/(reg[40]-1))
+Math.pow(reg[11]/reg[12],2/(reg[40]-1))
+Math.pow(reg[11]/reg[13],2/(reg[40]-1)))"/>
<!—-  compute U2 -->
<doOperation expression= "reg[42]=
1/(Math.pow(reg[12]/reg[11],2/(reg[40]-1))
+Math.pow(reg[12]/reg[12],2/(reg[40]-1))
+Math.pow(reg[12]/reg[13],2/(reg[40]-1)))"/>
<!—-  compute U3 -->
<doOperation expression= "reg[43]=
1/(Math.pow(reg[13]/reg[11],2/(reg[40]-
1))+Math.pow(reg[13]/reg[12],2/(reg[40]-
1))+Math.pow(reg[13]/reg[13],2/(reg[40]-1)))"/>
     </if>
<!-- If d1==0, U1=1, U2=0 et U3=0 -->
     <if test="reg[11]==0">
       <loadValue reg="41,42,43" value="1,0,0"></loadValue>
     </if>
  <!-- if d2==0, U1=1, U2=0 et U3=0 -->


       <if test="reg[12]==0">
         <loadValue reg="41,42,43" value="0,1,0"></loadValue>
      </if>
<!-- if d3==0, U1=1, U2=0 et U3=0 -->
      <if test="reg[13]==0">
        <loadValue reg="41,42,43" value="0,0,1"></loadValue>
      </if>
```

b) Local objective function computation and centroid determination

```
For (c = 1 to C)
 {
```

- Each PE computes terms : T1(c)=**U(c)^m , T2(c)=U(c)^m * data, T3(c)= U(c)^m * Distance(c)²**
- Each PE computes its local objective function (J)
- All PEs participate in the parallel hierarchical summation to compute global objective function J. This summation procedure was detailed in [27], it has a complexity of ***O(Log₂(n)) iterations***. The Host PE will retrieve and retain the result of the summation (J).

}

```
<!-- computing reg[47]= U1^m  -->
    <doOperation expression="reg[47]=Math.pow(reg[41],reg[40])"/>
<!-- computing reg[48]= U2^m  -->
   <doOperation expression="reg[48]=Math.pow(reg[42],reg[40])"/>
<!-- computing reg[49]= U3^m  -->
    <doOperation expression="reg[49]=Math.pow(reg[43],reg[40])"/>

<!-- computing reg[44]=U1^m * data  -->
    <doOperation expression="reg[44]=reg[47]*reg[0]"/>
<!-- computing reg[45]=U2^m * data  -->
    <doOperation expression="reg[45]=reg[48]*reg[0]"/>
<!-- computing reg[46]=U3^m * data  -->

    <doOperation expression="reg[46]=reg[49]*reg[0]"/>
<!-- computing reg[50]=U1^m * d1²  -->
    <doOperation expression="reg[50]=reg[47]*reg[11]*reg[11]"/>
<!-- computing reg[51]=U2^m * d2²  -->
    <doOperation expression="reg[51]=reg[48]*reg[12]*reg[12]"/>
<!-- computing reg[52]=U3^m * d3²  -->
    <doOperation expression="reg[52]=reg[49]*reg[13]*reg[13]"/>
<!--
All the marked PEs will participate to the hierarchical sum. The
result of this sum is stored in the host.
Hregs[41] contains the cardinality of C1 class.
Hregs[42] contains the cardinality of C2 class.
Hregs[43] contains the cardinality of C3 class.

Hregs[44] contains the sum of each PE term : (U1^m * data).
Hregs[45] contains the sum of each PE term : (U2^m * data).
Hregs[46] contains the sum of each PE term : (U3^m * data).

Hregs[47] contains the sum of each PE term : (U1^m).
Hregs[48] contains the sum of each PE term : (U2^m).
```

```
Hregs[49] contains the sum of each PE term : (U3^m).

Hregs[50] contains the sum of each PE term : (U1^m * d1²).
Hregs[51] contains the sum of each PE term : (U2^m * d2²).
Hregs[52] contains the sum of each PE term : (U3^m * d3²).
-->
    <doHSum PEregs="41,42,43,44,45,46,47,48,49,50,51,52"
Hregs="41,42,43,44,45,46,47,48,49,50,51,52"/>
</for-eachPE>
```

3.2.2.4. Class center computation and global objective function

```
<!--
  The host computes the global cost function J, the three new
class centers and sets up the iteration counter.
reg[0] contains the new value of C1 class.
reg[1]the new value of C2 class.
reg[2]the new value of C3 class.

reg[3]contains the value of Jn-1
reg[4]contains the value of Jn
-->
<host>
```

<!-- reg[i] contains the new value of Ci class = $V_i = \frac{\sum_{j=1}^{N} u_{ij}{}^{m} x_j}{\sum_{j=1}^{N} u_{ij}{}^{m}}$ -->

```
      <doOperation expression="reg[0]=reg[44]/reg[47]"/>
      <doOperation expression="reg[1]=reg[45]/reg[48]"/>
      <doOperation expression="reg[2]=reg[46]/reg[49]"/>
```

<!-- storing in reg[3] the last value of objective function J_{n-1}-->

```
      <doOperation expression="reg[3]=reg[4]"/>
```

<!-- computing the new value of objective function J_n -->

```
      <doOperation expression="reg[4]=reg[50]+reg[51]+reg[52]"/>
```

<!-- storing the new value of threshold reg[30]=J_n-J_{n-1} -->

```
      <doOperation expression="reg[30]=Math.abs(reg[4]-reg[3])"/>
<!-- incrementing the counter  -->
      <doOperation expression="reg[31]=reg[31]+1"/>
     </host>
 </doWhile>
```

3.2.2.5. Loop stopping test

The host calculates the absolute value $\left|J_{\text{n}} - J_{n-1}\right|$ and compares it with an arbitrary threshold S_{th}.

- If $\left|J_{\text{n}} - J_{n-1}\right| < S_{th}$ then go to the end of procedure.
- Else, return back to the procedure of new class centers computation.

```
<doWhile test="reg[30]>reg[5]" target="host">
```

3.2.2.6. Membership decision

```
<!--
  Labelling the different image components: assigning to each PE
the index of the class to which it belongs.
The registers reg[1],reg[2] and reg[3] of the mesh contains the
images of the first, the second and the third classes
respectively.
-->
<for-eachPE test="reg[0]!=0">
  <!--if(minimum(d1,d2,d3)==d1) reg[1]= C1 class -->
  <if test="min(reg[8], reg[9], reg[10])==reg[8]">
   <loadValue reg="1" value="host.reg[0]"/>
  </if>
  <!--if(minimum(d1,d2,d3)==d2) reg[1]= C2 class -->
  <if test="min(reg[8], reg[9], reg[10])==reg[9]">
   <loadValue reg="2" value="host.reg[1]"/>
  </if>
 <!--if(minimum(d1,d2,d3)==d3) reg[1]= C2 class -->
  <if test="min(reg[8], reg[9], reg[10])==reg[10]">
   <loadValue reg="3" value="host.reg[2]"/>
  </if>
  </for-eachPE>
<!--
  End of the program
-->
</prog>
```

4. Materials And Results

4.1. Materials:

After presenting the proposed PFCM and its corresponding program code, we have to evaluate the memory space required by the used variables and parameters. As we have seen in section 3), our program requires some physical components inside each PE of the RMC matrix and others inside the host PE.

The host PE uses some registers where it stores the required data about the input image and the instruction program that it sends to the PEs matrix in a SIMD manner. Also, each PE uses its own registers to carry out the operations coming from the host. In the same way, these operations need the internal registers inside each PE according to the image input and to the temporary local variables to execute all the instructions of the PFCM. Thus the total memory size required for each matrix PE equals 78 bytes, while, for the host PE, this total memory size required equals 63 bytes. The size and the contents of this required memory is detailed in the following table.

Table 1

Required registers of the RMC during the PFCM program execution

Register name	**Content**	**Target PE**	**Memory size (byte)**
reg[0], reg[1] and reg[2]	Values of $C_{1,}$ C2 and C3 class center	Host PE	3
reg[3], reg[4] and reg[5]	Values of J_{n-1} , J_n and S_{th}	Host PE	12
reg[0]	Gray level of the pixel	Matrix PE	1
reg[40]	Fuzzyfication parameter m	Matrix PE	1
reg[30]	Absolute value of (J_n-J_{n-1})	Matrix PE	4
reg[8], reg[9] and reg[10]	Values of the 3 class centers	Matrix PE	12
reg[11], reg[12] and reg[13]	Distances d1, d2, d3 from the PE to the 3 class centers	Matrix PE	12
reg[41], reg[42] and reg[43]	Membership degrees U1,U2 and U3	Matrix PE	12
reg[44], reg[45] and reg[46]	$Data*U1^m$, $Data*U2^m$, $Data*U3^m$	Matrix PE	12
reg[47], reg[48] and reg[49]	$U1^m$, $U2^m$, $U3^m$	Matrix PE	12
reg[50], reg[51] and reg[52]	$d1^2*U1^m$, $d2^2*U2^m$, $d3^2*U3^m$	Matrix PE	12
reg[41], reg[42], reg[43]	sum(U1),sum(U2) and sum(U3)	Host PE	12
reg[44], reg[45], reg[46]	$sum(Data*U1^m)$,$sum(Data*U2^m)$ and $sum(Data*U3^m)$	Host PE	12
reg[47], reg[48], reg[49]	$sum(U1^m)$,$sum(U2^m)$ and $sum(U3^m)$	Host PE	12
reg[50], reg[51], reg[52]	$sum(d1^2*U1^m)$,$sum(d2^2*U2^m)$ and $sum(d3^2*U3^m)$	Host PE	12

4.2. Program results:

The input MRI image is the one of figure 3a); this image will be segmented to sort out its three components. The execution of the presented parallel program leads to the following results: The image of figure 3a) corresponds to a human brain slice, it is the original input image of the program. Figures 3b), 3c) and 3d) represent the three matters of the brain. They are named respectively the grey matter, cerebrospinal fluid and white matter.

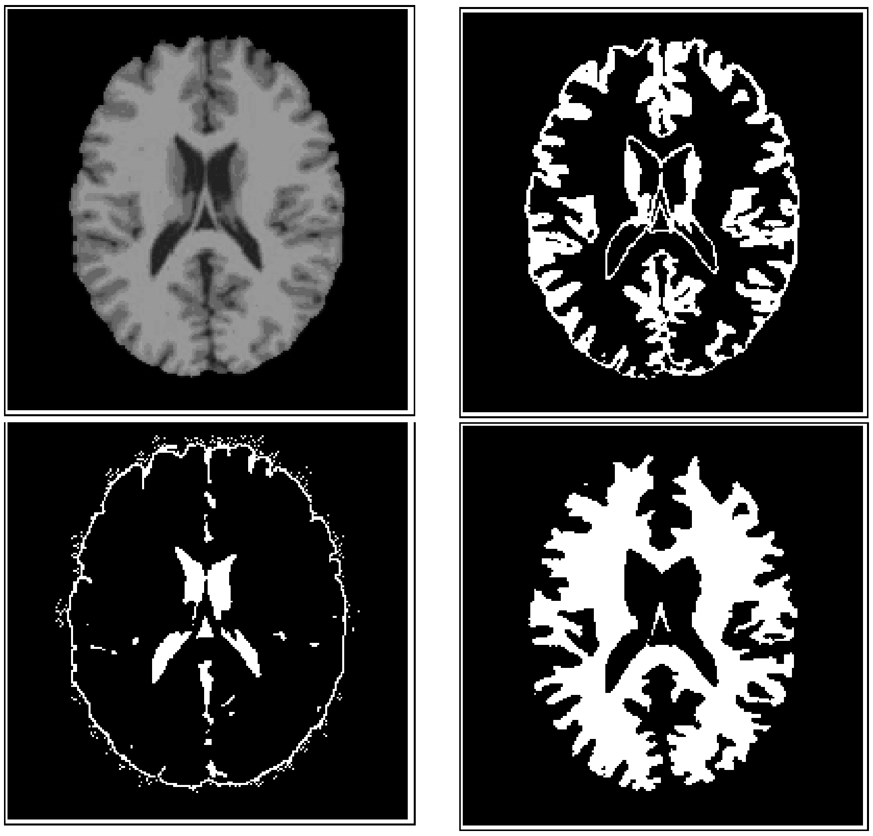


Figure 3. PFCM segmentation of an MRI brain image

In order to evaluate the effectiveness features of the proposed program, we focused the study on the dynamic convergence analysis of the method. To do so, we present four cases of study. For each case we use the same input MRI image, but the initial class centers are changed. The obtained results are presented as follows:

In the first case, the initial class centers are arbitrarily chosen as: (c_1, c_2, c_3) = (1, 30, 255). The algorithm converges to the final class centers (c_1, c_2, c_3) = (27.96, 102.39, 147.53) after 14 iterations, as in table 2.

Table 2. Different states of the parallel fuzzy classification algorithm starting from class centers (c_1, c_2, c_3) = (1, 30, 255).

Iteration	Value of each class center			Absolute value of the Error
	c1	c2	c3	$\lvert J_n - J_{n-1} \rvert$
1	1,00	30,00	255,00	69909116,17
2	75,11	96,74	146,36	65182804,53
3	53,90	99,71	146,63	1572217,07
4	39,55	102,08	147,02	666804,12
5	32,87	102,74	147,41	134784,03
6	30,04	102,78	147,57	22837,15
7	28,87	102,68	147,60	3962,85

8	28,37	102,58	147,59	790,03
9	28,16	102,51	147,57	189,27
10	28,05	102,46	147,56	52,75
11	28,01	102,43	147,55	16,07
12	27,98	102,41	147,54	5,12
13	27,97	102,40	147,54	1,66
14	27,96	102,39	147,54	0,1

In the second case, the initial class centers are arbitrarily chosen as: (c1, c2, c3) = (1, 2, 3). Table 3, shows that the algorithm converges to the same final class centers as in the first case, (c1, c2, c3) = (27.96, 102.39, 147.53), after 21 iterations.

Table 3. Different states of the parallel fuzzy classification algorithm starting from class centers (c_1, c_2, c_3) = (1, 2, 3)

Iteration	Value of each class center			Absolute value of the Error
	C_1	C_3	C_2	$\left\|J_n - J_{n-1}\right\|$
1	1,00	2,00	3,00	96142309,83
2	103,37	118,34	113,33	86051891,68
3	97,67	129,09	124,34	1672751,62
4	89,09	138,00	127,86	1348767,60
5	80,14	145,70	124,04	1362933,52
6	68,09	149,80	119,12	1250012,47
7	53,42	150,18	114,61	1149554,44
8	42,00	149,79	110,60	629238,18
9	35,37	149,18	107,59	233335,30
10	31,90	148,63	105,55	78947,64
11	30,10	148,22	104,27	26683,02
12	29,14	147,95	103,48	9012,82
13	28,62	147,78	103,02	3024,55
14	28,33	147,68	102,75	1007,82
15	28,17	147,61	102,60	334,00
16	28,07	147,58	102,51	110,30
17	28,02	147,56	102,45	36,34
18	27,99	147,55	102,42	11,96
19	27,97	147,54	102,41	3,93
20	27,97	147,54	102,40	1,29
21	27,96	147,53	102,39	0,1

In the third case, the initial class centers are arbitrarily chosen as: (c_1, c_2, c_3) = (140, 149, 150). Table 4, shows that the algorithm converges to the same final class centers as in the first case, (c_1, c_2, c_3) = (27.96, 102.39, 147.53), after 17 iterations.

Table 4. Different states of the parallel fuzzy classification algorithm starting from class centers (c_1, c_2, c_3) = (140, 149, 150).

Iteration	Value of each class center			Absolute value of the Error $\left\|J_n - J_{n-1}\right\|$
	c1	c2	c3	
1	140,00	149,00	150,00	15560879,96
2	111,26	119,70	140,72	6732394,85
3	88,45	109,02	145,64	2869304,41
4	67,66	104,62	147,64	1908707,56
5	48,77	104,28	147,85	1224690,56
6	37,57	104,20	147,90	392855,00
7	32,31	103,76	147,89	84405,72
8	29,97	103,30	147,81	17836,70
9	28,93	102,96	147,73	4301,86
10	28,44	102,73	147,66	1196,10
11	28,21	102,59	147,61	364,36
12	28,09	102,50	147,58	116,07
13	28,03	102,45	147,56	37,68
14	27,99	102,42	147,55	12,33
15	27,98	102,41	147,54	4,04
16	27,97	102,40	147,54	1,33
17	27,96	102,39	147,53	0,1

In the fourth case, the initial class centers are chosen, using the a preprocessing parallel histogram computation procedure of [25, 26] that orients the class centers towards the histogram modes of the image. In this case, the initial values of the class centers are: (c_1, c_2, c_3) = (23,102,150). For this case, we notice that the algorithm converges to the same final class centers as in the first case, (c_1, c_2, c_3) = (27.96, 102.39, 147.53), after only 8 iterations. See Table 5.

Table 5. Different states of the parallel fuzzy classification algorithm starting from class centers (c_1, c_2, c_3) = (23, 102, 150).

Iteration	Value of each class center			Absolute value of the Error $\left\|J_n - J_{n-1}\right\|$
	c1	c2	c3	
1	23,00	102,00	150,00	2399103,06
2	25,98	102,51	148,00	68294,83
3	27,19	102,43	147,65	5373,00
4	27,65	102,39	147,56	675,20
5	27,83	102,37	147,54	92,75
6	27,90	102,37	147,53	13,68
7	27,93	102,38	147,53	2,27
8	27,94	102,38	147,53	0,05

To illustrate this analysis we use the following figures 4, 5, 6 and 7, to show the curves of the different data, respectively, of Tables 2, 3, 4 and 5. Theses curves represent the dynamic changes of each class center.

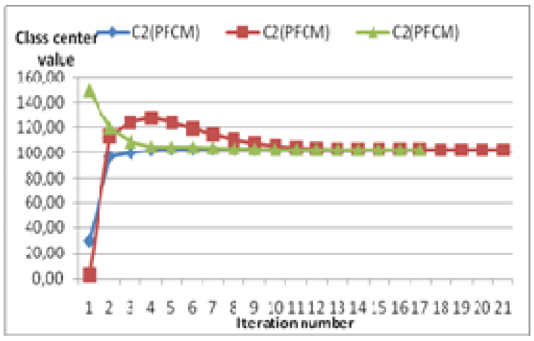


Figure 4. Case 1: Dynamic changes of the class centres starting from values (c1,c2, c3) = (1, 30, 255)

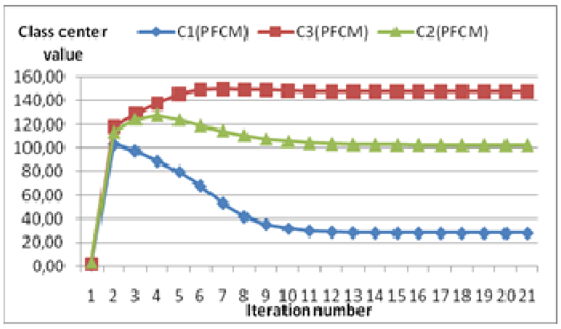


Figure 5. Case 2: Dynamic changes of the class centres starting from values (c1,c2, c3) = (1, 2, 3) .

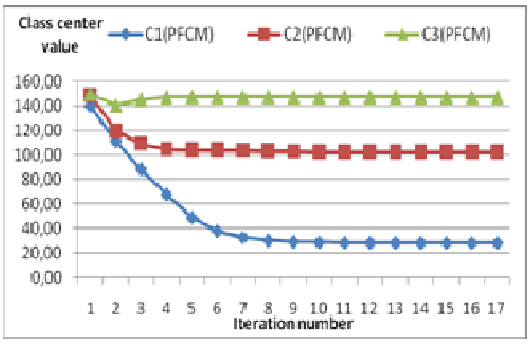


Figure 6. Case 3: Dynamic changes of the class centres starting from values (c1,c2, c3) = (140,149, 150)

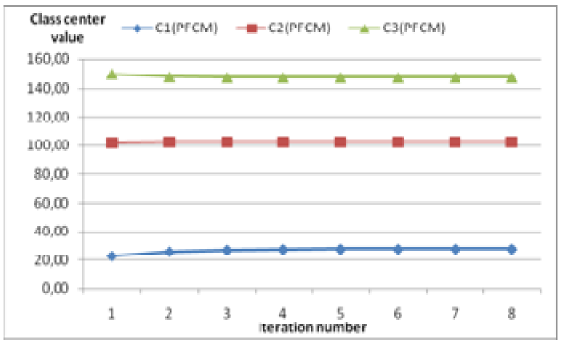


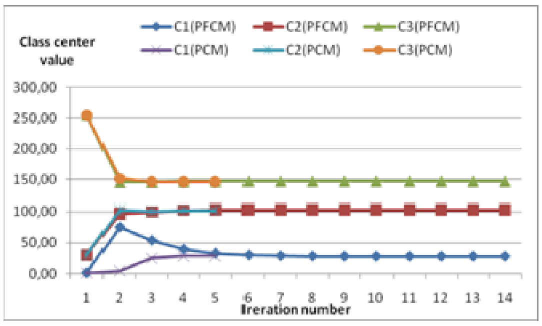
Figure 7. Case 4: Dynamic changes of the class centres starting from values (c1,c2, c3) = (23, 102,150)

Through the obtained results of the above cases of study, we can see that the algorithm converges to the same class centers. Its complexity in terms of iteration number depends on the initial values of the class centers.

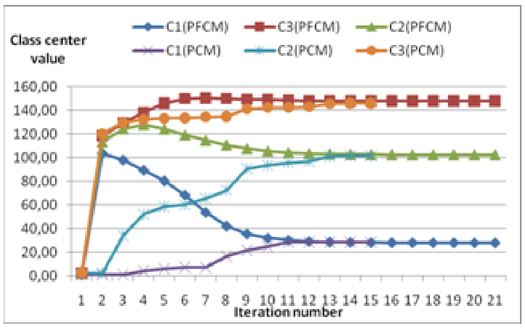
In order to evaluate the properties of the proposed algorithms, we have to compare the obtained results to other former works proposed in [27] for PCM algorithm.

In Figure 8, we start a comparison between the obtained results of the proposed "Parallel Fuzzy C-means" and parallel C-means of [27], For the same image and the same initialization parameters. We notice the following remarks:

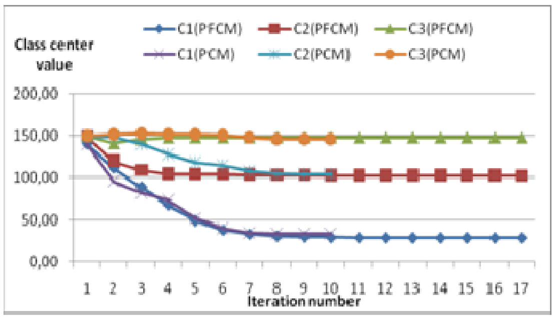
- With PFCM, final values of the cluster centers are exactly the same regardless of the initial values cluster centers. While in the case of PCM, we notify some differences between final cluster centers. This leads to conclude that the PCM algorithm exits the classification procedure even if it has not yet reached the real centers. Thus, we can say that PFCM is more accurate than PCM.
- For example, in the case of the first initialization, PCM leaves the classification in the fifth iteration, but PFCM continues to seek more accurate information until the 14th iteration.



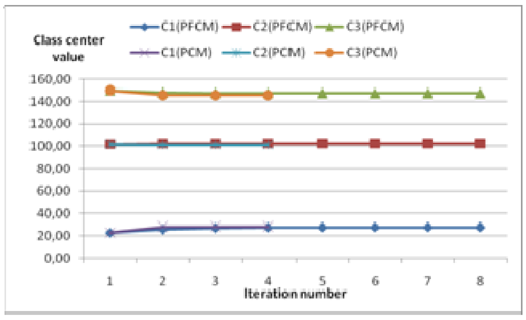
a) PFCM and PCM Class centers of case 1



b) PFCM and PCM Class centers of case 2.



c) PFCM and PCM Class centers of case 3.



c) PFCM and PCM Class centers of case 4.

Figure 8. Dynamic changes for different initialisations of class centers for PFCM and PCM algorithms.

5. Complexity Analysis

In order to evaluate the complexity of our PFCM algorithm, it is useful to report the complexities of all its stages. They are summarized in the following table 6.

Table 6. The complexities of each stage of the proposed parallel algorithm

Stage	Time Complexity
1- Data broadcasting	$O(1) + O(c)$
2- Distance computation	$O(c) + O(\log_2 c)$
3- Membership decision	$O(1)$
4- Objective function computation	$O(\log_2 c) + O(c.\log_2(N))$
5- Loop stop test	$O(1)$
6- New class center determination	$O(c) + O(c.\log_2(N))$
The complexity of the proposed algorithm	$O(c) + O(\log_2 c) + O(c.\log_2(N))$

The global complexity of our PFCM is: $O(c + \log_2 c + c.\log_2(N))$ *times*

If we consider a number of M features for each data point, the number M appears in some steps of the algorithm. Hence, this algorithm can be extended easily for any (M>1) features. In this case the complexity becomes:

$$O(Mc + \log_2 c + c.\log_2(N))times \approx \quad O(Mc + c\log_2(N))times$$

While for the PCM algorithm the complexity is: $O(Mc + c.\log_2(N/c))times$.

In this paper, the input data set of the algorithm is a gray leveled image, where each point has M=1 feature (its gray level) as in [27].

As shown in table 7, we can finally conclude that the proposed PFCM algorithm is more accurate than the PCM one. But its complexity remains greater than the one obtained in [27] and equal to these in [28]

Table 7. Comparison of complexities for parallel C-mean and Fuzzy C-mean algorithms.

Reference	Algo.	Architecture	Bus width (bit)	Processors	Time complexity
O.Bouattane [27]	PCM	RMESH	$O(\log_2 N)$	N	$O(k.M + \log_2 (k.(N/k)^k)$
Jenq [28]	PCM	RMESH	$O(\log_2 N)$	N	$O(kM + k \log_2 N)$
This paper	PFCM	RMESH	$O(\log_2 N)$	N	$O(k.M + k \log_2 N)$

6. CONCLUSION

In this paper, we have presented a method for parallelizing the fuzzy c-means classification algorithm and its implementation on a massively parallel reconfigurable mesh computer. An application of this algorithm to the MRI images segmentation was considered. The elaborated program was performed on the reconfigurable mesh computer emulator. The obtained results show that, in little number of iterations, the algorithm converges to the same final class centers what ever their starting values. This is to proof its accuracy comparing to the well known C-mean algorithms. Hence, the parallel computation method is proposed essentially to reduce the complexity of the fuzzy clustering algorithms. Also it was shown that to enhance the effectiveness of this work, it is useful to improve the complexity of this algorithm by avoiding random initializations of the class centers.

REFERENCES

[1]. S.Murugalavalli and V.Rajamani. A High Speed parallel Fuzzy C-Mean algorithm for brain tumor segmentation. BIME Journal, Volume (06), Issue (1), Dec. 2006, pp. 29-34.

[2]. S.Rahimi, M.Zargham, A. Thakre and D. Chillar. A Parrallel Fuzzy C-Mean Algorithm for Image Segmentation. Proc. of the IEEE Conf. 0-7803-8376-1/04/. 2004 pp. 234- 237.

[3]. B.Monien and R.Feldman . Parallel Fuzzy C-means Clustering for Large Data Sets. (Eds) EURO-Par (2002) LNCS 2400. Springer Verlag Heidelberg, pp. 365-374.

[4]. M. Cannataro, A. Congiusta, A. Pugliese, D. Talia, P. Trunfio (2004). Distributed data mining on grids: services, tools, and applications. IEEE Transactions on Systems, Man and Cybernetics, Part B, vol. 34, no. 6, pp. 2451 – 2465.

[5]. K. Kubota, A. Nakase, H. Sakai and S. Oyanagi (2000). Parallelization of decision tree algorithm and its performance evaluation. Proceedings of the Fourth International Conference on High Performance Computing in the Asia-Pacific Region, vol. 2, pp. 574-579.

[6]. [M. W. Kim, J. G. Lee and C. Min (1999). Efficient fuzzy rule generation based on fuzzy decision tree for data mining. Proceedings of the IEEE International Fuzzy Systems Conference FUZZ-IEEE '99. pp 1223 – 1228.

[7]. A. Evsukoff, M. C. A. Costa and N. F. F. Ebecken (2004). Parallel Implementation of Fuzzy Rule Based Classifier. Proceedings of the VECPAR'2004, vol. 2, pp. 443-452.

[8]. P. K. H. Phua and D. Ming. (2003). Parallel nonlinear optimization techniques for training neural networks. IEEE Transactions on Neural Networks, vol. 14, no. 6, pp. 1460 - 1468.

[9]. M. C. A. Costa and N. F. F. Ebecken (2001). A Neural Network Implementation for Data Mining High Performance Computing. Proceedings of the V Brazilian Conference on Neural Networks, pp. 139-142.

[10]. R. Agrawal and J. C. Shafer (1996).Parallel mining of association rules. IEEE Transactions on Knowledge and Data Engineering, vol. 8, no. 6, pp. 962 - 969.

[11]. L. Shen, H. Shen and L. Cheng (1999). New algorithms for effcient mining of association rules. Information Sciences 118, pp. 251 – 268.

[12]. B. Boutsinas and T. Gnardellis (2002). On distributing the clustering process. Pattern Recognition Letters 23, pp. 999–1008.

[13]. J.Elmesbahi, O.Bouattane, M. Sabri and M. Chaibi. A Fast Algorithm for Ranking and Perimeter Computation on a Reconfigurable Mesh Computer. Proceedings of the IEEE international conference on Systems Man and Cybernetics, october 2-5. Texas. (1994) pp 1898-1902.

[14]. J. B. MacQueen, “Some Methods for classification and Analysis of Multivariate Observations”, Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, 1:281-297,1967.

[15]. J.C. Bezdek, “Pattern Recognition with Fuzzy Objective Function Algorithms”, Plenum Press, New York 1981.

[16]. Dzung L. Pham,Jerry L. Prince. 1999. “An adaptative fuzzy c means algorithm for image segmentation in the presence of intensity inhomogeneities”. *In Pattern recognition letters* 20 p. 57-68.

[17]. Ma L. and Staunton R.C., 2007. “A modified fuzzy C-means image segmentation algorithm for use with uneven illumination patterns Pattern Recognition, Vol. 40, pp. 3005-3011.

[18]. Guo Y., Cheng H.D., Zaho W. and Zhang Y. 2008. “A Novel Image Segmentation Algorithm Based on Fuzzy C-means Algorithm and Neutrosophic Set”. Proceeding of the 11th joint conference on information sciences. Atlantis Press.

[19]. H.D. Heng, X.H. Jiang, Y. Sun, J. Wang, Color image segmentation: advances and prosoects, Pattern Recognition 34 (2001) 2259–2281.

[20]. A. Zijdenbos, B.M. Dawant, Brain segmentation and white matter lesion detection in MR images, Critical Reviews in Biomedical Engineering 22 (5/6) (1994) 401–465.

[21]. J.H. Chang, K.C. Fan, Y.L. Chang, Multi-modal gray-level histogram modeling and decomposition, Image and Vision Computing 20 (2002) 203–216.

[22]. Dunn J.C., “A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters”. Journal of Cybernetics 3(3), 1973, pp. 32–57.

[23]. M. Youssfi, O. Bouattane, and M.O. Bensalah. ” On the object modelling of the Massively parallel architecture Computers”. In the proceeding of IASTED International conference of Software Ingineering (SE 2010). Pages 71-78, Innsbruck, Autriche, February 16-18, 2010.

[24]. M. Youssfi, O. Bouattane, M.O. Bensalah, “A massively parallel reconfigurable mesh computer emulator: design, modeling and realization”. Journal of Software Engineering and Applications volume 3, p 11–26. 2010.

[25]. M. Eshaghian-Wilner Mary, R. Miller, The systolic reconfigurable mesh, Parallel Processing Letters 14 (3–4) (2004) 337–350. ISSN 0129-6264.

[26]. J. Jang, H. Park, V.K. Prasanna, A fast algorithm for computing histograms on a reconfigurable mesh, in: Fourth Symposium on the Frontiers of Massively Parallel Computation, 19–21 October 1992, pp. 244–251.

[27]. O. Bouattane, B. Cherradi, M. Youssfi and M.O. Bensalah. “Parallel c means algorithm for image segmentation on a reconfigurable mesh computer”. Parallel Computing, Volume 37, Issues 4-5, Pages 230-243, April-May 2011, ELSEVIER.

[28]. J.F. Jenq, S. Sahni. Reconfigurable mesh algorithms for image shrinking, expanding, clustering and template matching. International parallel processing symposium, (1991) pp.208–215.