BANDWIDTH ADAPTIVE HARDWARE ARCHITECTURE OF K-MEANS CLUSTERING FOR INTELLIGENT VIDEO PROCESSING

Tse-Wei Chen and Shao-Yi Chien

Media IC and System Lab
Graduate Institute of Electronics Engineering and Department of Electrical Engineering
National Taiwan University
BL-421, No. 1, Sec. 4, Roosevelt Rd., Taipei 106, Taiwan
{variant, sychien}@media.ee.ntu.edu.tw

ABSTRACT

K-Means is a clustering algorithm that is widely applied in many fields, including pattern classification and multimedia analysis. Due to real-time requirements and computational-cost constraints in embedded systems, it is necessary to accelerate K-Means algorithm by hardware implementations in SoC environments, where the bandwidth of the system bus is strictly limited. In this paper, a bandwidth adaptive hardware architecture of K-Means clustering is proposed. Experiments show that the proposed hardware has the maximum clock speed 400MHz with TSMC 90nm technology, and it can deal with feature vectors with different dimensions using five parallel modes to utilize the input bandwidth efficiently.

Index Terms— K-Means, clustering methods, pattern recognition, parallel architectures, hardware design.

1. INTRODUCTION

Clustering is a fundamental and important technique in unsupervised machine learning and data mining, and K-Means is a well known algorithm that is suitable to many kinds of multimedia applications, including image and video segmentation [1, 2]. Many issues of K-Means, including the initialization of centroids [3] and acceleration of speed [4], are explored. Besides, due to real-time requirements of K-Means for image segmentation and video imaging, many kinds of hardware acceleration schemes are also proposed [5, 6]. There are also methods using filtering algorithm and KD-trees for FPGA implementation [7] and software/hardware co-design techniques [8]. The hardware design of K-Means clustering has received more and more attention recently.

In our previous work, a Silicon Intellectual Property (SIP) of K-Means algorithm is proposed to accelerate image segmentation in SoC environments for embedded systems [9]. The K-Means SIP can handle five dimensional feature vectors efficiently by using parallel processing elements, and the performance of image segmentation is tremendously higher than other processors. However, when the number of dimensions is smaller than five, the system bandwidth and processing elements become wasted since these resources are not utilized well. The bandwidth issue is significant for K-Means SIP since the same input vectors needs to fed to the hardware iteratively, consuming large amount of bandwidth resources. Therefore, a new K-Means clustering hardware architecture, which employs bandwidth adaptive mechanism, is proposed in this paper. Not only does it have much higher flexibility and robustness, but it also achieves better performances than the previous work. In addition, both Manhattan distance and Euclidean distance can be selected for the distance measurement by using the same hardware architecture. Furthermore, there are a total of five parallel modes to configure the processing elements and to allocate feature vectors to different computing resources, and the random initialization method is implemented in the hardware modules to accelerate the processing speed and to find appropriate solutions for clustering.

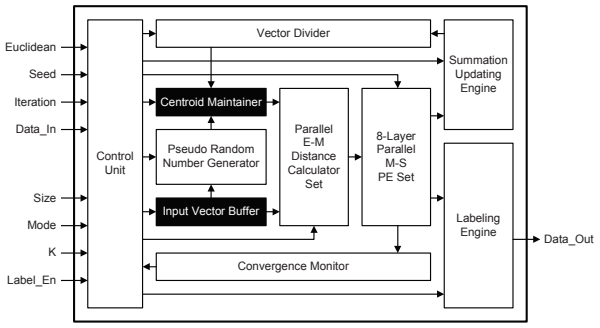

Fig. 1. An overview of the proposed architecture.

2. K-MEANS ALGORITHM AND HARDWARE CONSIDERATIONS

K-Means clustering algorithm regards the i-th input data $\boldsymbol{x}_i$ as a D-dimensional feature vector, which is represented as

$$\boldsymbol{x}_i = (x_{i,1}, \ldots, x_{i,D}), \tag{1}$$

where D is the number of vector dimensions. Owing to hardware-cost constraints in embedded systems, the proposed hardware is designed to deal with clustering problems with feature vectors whose dimension number $D \leq 16$. The iterative steps of K-Means clustering algorithm used in the proposed hardware are stated as follows.

Step 1: According to the cluster number K, K vectors are randomly selected to be centroids of K clusters. Each centroid is then represented as $\boldsymbol{\mu}_k^{(0)}$, where k stands for the k-th centroid. The hardware needs to read all the input vectors before randomly determining K centroids. This approach, also called Forgy initialization [3], is friendly to hardware design.

Table 1. Five Parallel Modes of the Proposed Hardware

Parallel Mode	Mode-A	Mode-B	Mode-C	Mode-D	Mode-E
Vector Dimension	1	1 – 2	1 – 4	1 – 8	1 – 16
Processing Speed	×16	×8	×4	×2	×1

Step 2: Each input vector is then assigned to its corresponding cluster according to the nearest mean function, which is defined as

$$\phi^{(t)}(k|\boldsymbol{x}_i) = \begin{cases} 1, & \text{if } k = \arg\min_j D(\boldsymbol{x}_i, \boldsymbol{\mu}_j^{(t)}) \\ 0, & \text{otherwise.} \end{cases}, \quad (2)$$

where $\boldsymbol{\mu}_j^{(t)}$ denotes the centroid of the j-th cluster in the t-th iteration, and $D(.)$ is the distance measurement. Both Manhattan distance and Euclidean distance can be selected for distance measurement by users.

Step 3: To update the centroid of each cluster, the following equation is used:

$$\boldsymbol{\mu}_k^{(t+1)} = \frac{\sum_{i=1}^{N} \phi^{(t)}(k|\boldsymbol{x}_i)\boldsymbol{x}_i}{\sum_{i=1}^{N} \phi^{(t)}(k|\boldsymbol{x}_i)}, \quad (3)$$

where N is the total number of input vectors.

Step 4: Whether the iteration should stop or not is determined according to the total distortion, which is defined as

$$\Delta^{(t)} = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} \phi^{(t)}(j|\boldsymbol{x}_i) D(\boldsymbol{x}_i, \boldsymbol{\mu}_j^{(t)}). \quad (4)$$

The iteration stops if the maximum number of iteration is reached or $|\Delta^{(t+1)} - \Delta^{(t)}| < Th_\Delta$, where Th_Δ is a small positive constant, and it continues to the next step. Otherwise, it will jump to step 2 for the $(t+1)$-th iteration.

Finally, each input vector is assigned to the nearest neighbor of cluster centroids, and all vectors are segmented to K clusters. Since this iteration method does not guarantee that the best clustering results will be found in the solution space, the total distortion in (4) can be used to evaluate the fitness of clustering results when the value of K is fixed. In addition to real-time requirements in multimedia applications, another purpose of accelerating K-Means using hardware is to obtain clustering results close to the global solution by efficiently repeating the algorithm with different initializations.

3. PROPOSED HARDWARE ARCHITECTURE

The proposed K-Means hardware is designed to work under the intelligent video processing platform where the bandwidth of bus is 128 bits. In addition to the proposed K-Means hardware, there are other Silicon Intellectual Properties (SIPs) connected to the bus to share the same on-chip memory. The bit-width of each dimension of the input vector is assumed to be 8-bit, and vectors with the maximum dimension number (16 dimensions) exactly fit the bandwidth of the 128-bit system bus. One of the main contribution of the proposed hardware, the bandwidth adaptive mechanism, is realized by five parallel modes for different vector dimensions. The details of each mode are shown in Table 1, and suitable modes can be selected to obtain the best performance. An overview of the proposed K-Means clustering hardware is illustrated in Fig. 1, and the functionality of each module will be explained in the following subsections.

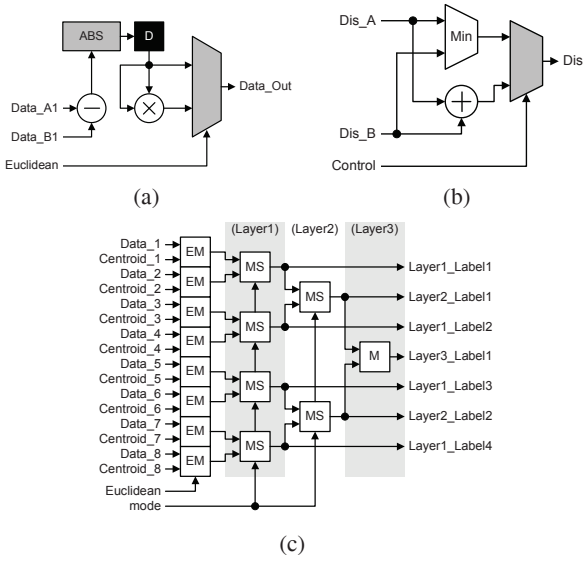

Fig. 2. (a) The architecture of "E-M distance calculator" module. (b) The architecture of "M-S Processing Element (M-S PE)" module. (c) The architecture of "Parallel E-M Distance Calculator Set" and "3-Layer Parallel M-S PE Set" module.

3.1. Control Unit and Pseudo Random Number Generator

"Control Unit" module controls all the components inside the K-Means hardware, managing the connection of input signals and modules. Initially, "Pseudo Random Number Generator" module generates K random numbers and stores them into a local buffer. When the input vectors are fed into "Control Unit" module, vectors whose input order is the same as any of the selected random number are saved as the initial centroids. K centroids will be initialized using this approach after all the input vectors are checked, and these centroids are stored in "Centroid Maintainer" module to be updated by iterative steps.

3.2. Parallel E-M Distance Calculator Set

"Parallel E-M Distance Calculator Set" contains 256 "E-M Distance Calculator" modules, which can calculate the distance of input vectors and 16 centroids. The main architecture of "E-M Distance Calculator" module is shown in Fig. 2(a), where the output distance can be selected to be Euclidean distance or Manhattan distance using the same hardware resources. When Euclidean distance is selected, the module computes the square of the difference of the input vector and the corresponding centroid; when Manhattan distance is selected, the module simply computes the absolute difference of the input vector and the corresponding centroid. Note that each "E-M Distance Calculator" computes the distance of only one dimension of vectors and centroids.

To explain the operation of "E-M Distance Calculator" in different modes, an example is shown in the left side of Fig. 2(c), which is a simple system that only deals with input vectors with dimension $D = 4$ and cluster number $K = 2$. In this example system, there are only three parallel modes. In the first mode, these modules deal with 1-dimensional vectors. In Fig. 2(c), "Data_1" and "Data_2" both contain the same 1-dimensional input vector; "Centroid_1" and

"Centroid_2" contain the 1-dimensional vector of the first centroid and the second centroid respectively. A total of eight "E-M distance calculator" modules are configured as four parallel processors which can compute the distance of four 1-dimensional input vectors to their corresponding centroids simultaneously. In the second mode, these modules deal with 2-dimensional vectors. In Fig. 2(c), "Data_1" and "Data_3" both contain the first dimension of the same 2-dimensional input vector, and "Data_2" and "Data_4" both contain the second dimension of the same 2-dimensional input vector; "Centroid_1" and "Centroid_2" contain two dimensions of 2-dimensional vector of the first centroid, and "Centroid_3" and "Centroid_4" contain the two dimensions of the 2-dimensional vector of the second centroid. Eight "E-M distance calculator" modules are configured as two parallel processors which can compute the distance of two 2-dimensional input vectors and their corresponding centroids simultaneously. Similarly, in the third mode, eight "E-M distance calculator" modules are configured as one processor to compute the distance of one 4-dimensional input vector and its corresponding centroid. Much more powerful than this example, the proposed architecture deals with input vectors with dimension $D = 16$ and cluster number $K = 16$, and there are five parallel modes as shown in Table 1.

3.3. 8-Layer Parallel M-S PE Set

"8-Layer Parallel M-S PE Set" module contains a set of tree-structured "M-S Processing Element (M-S PE)" modules. For a L-layer module, the number of "M-S PE" module is $2^L - 1$, and a 3-layer example is shown in Fig. 2(c), where there are seven "M-S PE" modules in three layers. The main architecture of "M-S PE" module is shown in Fig. 2(b), and "M-S" stands for "Minimum" and "Summation." This processing element can be configured to execute two kinds of operations, "Minimum" or "Summation." To explain the functionality of "8-Layer Parallel M-S PE Set" module in different modes, the example with three parallel modes in Fig. 2(c) is used. In the first mode, The operation of "M-S PE" in the first layer are set to "Minimum," and four labels, containing the nearest neighbor information of input vectors, are sent to the output signals. In the second mode, The operations of "M-S PE" modules in the first layer are set to "Summation," and the operations of "M-S PE" modules in the second layer are set to "Minimum." The distances of two pairs of 2-dimensional vectors are calculated in the first layer, and two labels, containing the nearest neighbor information of input vectors, are sent to the output signals. Similarly, in the third mode, the distance of a pair of 4-dimensional vectors is calculated in the first and the second layer, and finally one label is sent to the output signal.

Note that "M-S PE" modules in the last layer only has "Minimum" operation in this 3-layer example, and similarly, "M-S PE" modules in the last three layers only have "Minimum" operation in the proposed 8-layer architecture. In addition, in order to adjust the cluster number K, the redundant centroids are set to be inactive in these "M-S PE" modules, and inactive centroids are not chosen to be updated in the next stage.

3.4. Summation Updating Engine

The architecture of "Summation Updating Engine" module is illustrated in Fig. 3(a), where the input vector stored in "Input Vector Buffer" module and the label information from "8-Layer Parallel M-S PE Set" module are sent to "Centroid Selector" module. The functionality of "Centroid Selector" module is to update the summation and count of corresponding clusters, which are the numerator

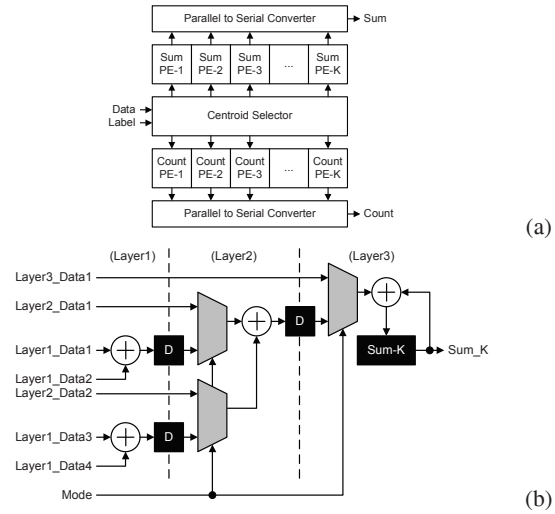

Fig. 3. (a) The architecture of "Summation Updating Engine" module. (b) The architecture of "Sum Processing Element (Sum PE)" module in "Summation Updating Engine" module.

and the denominator in (3) respectively. It distributes the updating information of each centroid to its corresponding unit according to different parallel modes. Modules which receive the updating information are "Sum Processing Element (Sum PE)" module and "Count Processing Element (Count PE)" module. Each centroid has one 16 "Sum PE" modules and one "Count PE" module. These two modules have similar architectures, and the architecture of "Sum PE" is shown in Fig. 3(b). Again, this is an example of the 3-layer system. The output data of different layers is connected to different stages in this module, and at most four input vectors can be added together when the dimension number is equivalent to one. The summation and count of each centroid are stored in a local buffer and connected to "Parallel to Serial Converter" module until "Vector Divider" module in the next stage requires data to perform divisions in (3).

3.5. Vector Divider

"Vector Divider" module contains 16 non-restoring bit-serial dividers, which are able to compute a division of a 16 dimensional vector in 10 cycles. The number of total cycles to compute new centroids depends on the number of cluster K. For instance, it takes 100 cycles to compute 10 cluster centroids of 16 dimensions when $K = 10$. Then new centroids are stored to "Centroid Maintainer" module for the next iteration. Moreover, a inspection mechanism is proposed to handle the situation when a centroid has no input vectors to be updated. To realize this mechanism, a comparator is integrated with "Vector Divider" module to check if the divisor is zero. When the divisor becomes zero, the output of "Vector Divider" is assigned to be the cluster centroid in the previous iteration. It stables the clustering procedure by eliminating the possibility of division by zero.

3.6. Convergence Monitor and Labeling Engine

"Convergence Monitor" module examines whether the iteration is finished according to (4) and returns the information to the "Control Unit" module. When the iteration is finished, the input vectors are

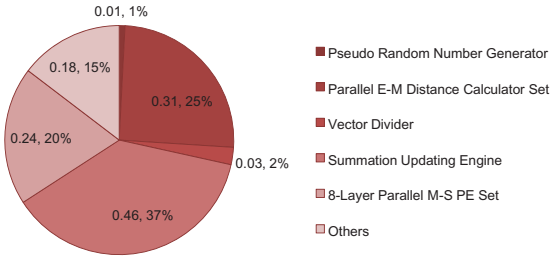

Fig. 4. The area (mm^2) and the percentage of each module of the proposed K-Means hardware.

fed to the hardware again to obtain the nearest neighbor of each input vector. The functionality of "Labeling Engine" module is to output the label or the vector of the nearest centroid of each input vector, and the latency and the maximum throughput of output data is 10 cycles and 128 bits/cycle respectively.

4. EXPERIMENTAL RESULTS

The experiments contain two parts. The first part is the architectural analysis. In this part, the critical path and hardware costs of each module in the proposed K-Means architecture are analyzed using Synopsys DesignVision with TSMC 90nm technology library. The maximum clock frequency is 400MHz, the total gate count is 440K, and the corresponding area is $1.23mm^2$. The area of each module is shown in Fig. 4, where computationally intensive processing units, such as "Parallel E-M Distance Calculator Set" module, "8-Layer Parallel M-S PE Set" module, and "Summation Updating Engine" module, occupy most of the area.

The second part is the comparison of hardware specifications. This work is compared to the previous work [9], and the results are summarized in Table 2. The proposed work supports higher dimensions of input vectors, achieves higher throughput, and has ability to handle more data number than the previous work. Since the proposed work uses TSMC 90nm technology, the area containing almost eight times of gate count of the previous work is less than three times of the area of the previous work. Moreover, the proposed work has five parallel modes for bandwidth adaptive mechanism to handle vectors with different dimensions efficiently. When the dimension number is equal to one, the processing speed of this work is 64 times faster than the previous work. In terms of the algorithms issue, the proposed work supports both Euclidean distance and Manhattan distance, applies random initialization approach, and includes the inspection mechanism which is able to stable the clustering process.

5. CONCLUSIONS AND FUTURE WORK

K-Means is an important clustering algorithm in the field of pattern recognition and data mining. To make this algorithm feasible for multimedia applications in real-time embedded systems, a bandwidth adaptive scheme for K-Means hardware is proposed. The architecture enables five parallel modes for different dimension numbers of feature vectors by configuring the processing elements. For future developments, modules for complicated distance measurement will be integrated with the existing architecture to construct a robust K-Means clustering engine.

Table 2. Comparison of This Work and The Previous Work

	ISCAS 2008 [9]	This Work
Technology	TSMC 0.18μm	TSMC 90nm
Clock Frequency	200MHz	400MHz
Gate Count	58K	440K
Area	$0.58mm^2$	$1.23mm^2$
Number of K	1 – 16	1 – 16
Vector Dimension	1 – 5	1 – 16
Parallel Mode	1 Fixed Mode	5 Modes
Throughput	2.5 Dimensions/cycle	16 Dimensions/cycle
Initialization Method	Predefined Centroid	Forgy [3]
Distance Measurement	Manhattan	Euclidean/Manhattan
Maximum Data Number	2^{17}	2^{20}

6. REFERENCES

[1] T.-W. Chen, S.-C. Hsu, and S.-Y. Chien, "Robust video object segmentation based on K-Means background clustering and watershed in ill-conditioned surveillance systems," in *Proceedings of IEEE International Conference on Multimedia and Expo*, July 2007, pp. 787–790.

[2] 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.

[3] J. M. Pena, J. A. Lozano, and P. Larranaga, "An empirical comparison of four initialization methods for the K-Means algorithm," *Pattern Recognition Letters*, vol. 20, no. 10, pp. 1027–1040, 1999.

[4] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "An efficient K-Means clustering algorithm: analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881–892, July 2002.

[5] M. Estlick, M. Leeser, J. Theiler, and J. J. Szymanski, "Algorithmic transformations in the implementation of K-Means clustering on reconfigurable hardware," in *Proceedings of the 2001 ACM/SIGDA ninth international symposium on Field programmable gate arrays*, 2001, pp. 103–110.

[6] T. Maruyama, "Real-time K-Means clustering for color images on reconfigurable hardware," in *Proceedings of International Conference on Pattern Recognition*, 2006, pp. 816–819.

[7] T. Saegusa and T. Maruyama, "An FPGA implementation of K-Means clustering for color images based on KD-Tree," in *Proceedings of International Conference on Field Programmable Logic and Applications*, Aug 2006, pp. 1–6.

[8] A. G. S. Filho, A. Frery, C. de Araujo, H. Alice, J. Cerqueira, J. Loureiro, M. de Lima, M. Oliveira, and M. Horta, "Hyperspectral images clustering on reconfigurable hardware using the K-Means algorithm," in *Proceedings 16th Symposium on Integrated Circuits and Systems Design*, Sep 2003, pp. 99–104.

[9] 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 *Proceedings of IEEE International Symposium on Circuits and Systems*, May 2008, pp. 2578–2581.