Power-Efficient Hardware Architecture of K-Means Clustering With Bayesian-Information-Criterion Processor for Multimedia Processing Applications

Tse-Wei Chen, *Member, IEEE*, Chih-Hao Sun, Hsiao-Hang Su, Shao-Yi Chien, *Member, IEEE*, Daisuke Deguchi, Ichiro Ide, and Hiroshi Murase, *Fellow, IEEE*

***Abstract*—A power-efficient K-Means hardware architecture that can automatically estimate the number of clusters in the clustering process is proposed. The contributions of this work include two main aspects. The first is the integration of the hierarchical data sampling in the hardware to accelerate the clustering speed. The second is the development of the "Bayesian-Information-Criterion (BIC) Processor" to estimate the number of clusters of K-Means. The architecture of the "BIC Processor" is designed based on the simplification of the BIC computations, and the precision of the logarithm function is also analyzed. The experiments show that the proposed architecture can be employed in different multimedia applications, such as motion segmentation and edge-adaptive noise reduction. Besides, the gate count of the hardware is 51 K with the 90-nm complimentary metal–oxide–semiconductor technology. It is also shown that this work can achieve high efficiency compared with a GPU, and the power consumption scales well with the number of clusters and the number of dimensions. The power consumption ranges between 10.72 and 12.95 mW in different modes when the operating frequency is 233 MHz.**

***Index Terms*—Clustering methods, energy efficiency, hardware design, K-Means, machine learning.**

I. Introduction

MULTIMEDIA processing is becoming a critical functionality for mobile devices. Since many kinds of acceleration methods of video coding and processing [1], [2] are developed for embedded computing, the real-time processing is possible for image data with different frame sizes. However, the computational complexity of machine learning algorithms [3], which are key techniques for semantic analysis of multimedia contents, is still high for resource-limited systems. As a key issue for implementation, low power consumption generally influences the design strategies. Therefore, the demands of low costs and energy efficiency turn the architectural design of machine learning hardware into a challenging issue for the development of mobile systems.

It is known that data clustering is an important part of machine learning algorithms, and K-Means and its variations [4], [5] are important clustering algorithms that have already been employed in different kinds of applications. In recent years, the development of K-Means hardware architectures with various requirements is receiving attention [6]–[11]. Maruyama *et al.* propose an FPGA implementation of real-time K-Means clustering for color images, employing a filtering algorithm for hardware implementation [9]. Ma *et al.* propose a real-time K-Means learning processor architecture, which can be extended to multiple-chip large-scale systems [10], [11]. Besides, in our previous works [12]–[15], the importance of hardware-oriented K-Means algorithms is emphasized, and mainly three kinds of architectures with different functionalities are reported. In our first work [12], the K-Means silicon intellectual property (SIP) that is designed for embedded systems is proposed. The architecture of the SIP is also regarded as the prototype of the following works. In our second work [13], the bandwidth adaptive K-Means architecture is proposed to handle feature vectors with different dimensions, and the processing elements are designed for the computations of both the Euclidean distance and the Manhattan distance. In our third work [15], the hierarchical K-Means (HK-Means) architecture is proposed to handle large numbers of clusters, and the gate count per cluster is the lowest among the three works. Although the second work and the third work provide solutions to different problems, not much attention is paid to the hardware cost since the previous works mainly focus on the functionality. Moreover, the estimation of numbers of clusters [16], which is one of the significant issues of K-Means, is not taken into the design considerations in our previous works [12], [13], [15] and related works [6]–[11], so the number of clusters has to be specified before the clustering process in these architectures.

In order to deal with the problems mentioned above, an efficient architecture that has low power dissipation and low area costs is proposed. In addition to the efficient design, three new strategies are adopted in this architecture. The first strategy is the hierarchical data sampling, which saves the computational costs of K-Means and speeds up the clustering process. The second strategy is the development of the "Bayesian-Information-Criterion (BIC) Processor," which can compute the BIC score [17], [18] and provides an option to estimate numbers of clusters. The

Manuscript received December 31, 2010; revised June 13, 2011; accepted July 20, 2011. Date of publication October 03, 2011; date of current version November 09, 2011. This work is supported by National Science Council in Taiwan under Contract NSC98-2917-I-002-113. This paper was recommended by Guest Editor V. Narayanan.

T.-W. Chen, D. Deguchi, I. Ide, and H. Murase are with Graduate School of Information Science, Nagoya University, Nagoya-shi 464-8601, Japan (e-mail: twchen@ieee.org).

T.-W. Chen, C.-H. Sun, H.-H. Su, and S.-Y. Chien are with Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan.

Color versions of one ore more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JETCAS.2011.2165231

third strategy is the design of the "Trial Monitor," which enables the trial computation of K-Means to achieve higher clustering quality than previous works. Although the low-cost property might come with low processing capacity compared with our previous works [12], [13], [15], multiple trials and the estimation of numbers of clusters can be completed without consuming the bandwidth of the system bus. In addition, the proposed efficient K-Means architecture can be applied to different multimedia applications, including image segmentation, color clustering, motion segmentation and edge-adaptive noise reduction. The low-cost specifications are capable of handling the target applications. Furthermore, the specifications of the efficient K-Means architecture are compared with the GPU architecture, and the scalability of power dissipation in the architecture is analyzed with different numbers of clusters and different numbers of dimensions.

The paper is organized as follows. First, the hierarchical sampling technique that is suitable for the efficient K-Means hardware is introduced in Section II. Then, the K-Means architecture and the "BIC Processor" are described in Sections III and IV, respectively. Next, the applications of the efficient K-Means hardware and the experimental results are shown in Section V. Finally, this paper is concluded in Section VI.

II. K-Means With Hierarchical Data Sampling

To design an energy-efficient K-Means hardware, it is necessary to reduce the time for the clustering process. In our previous work, the hierarchical K-Means algorithm [15] is applied to the hardware design. It is a method to generate tree structures for centroids of clusters, so that large numbers of clusters can be effectively handled. Different from our previous work, a hierarchical sampling technique that is suitable for the proposed K-Means hardware is introduced in this section.

Data sampling is a technique that can be applied to K-Means clustering for speed acceleration [9], [19]. Here, an example of using the sampling technique is explained. First, the K-Means is performed with some samples of input vectors. Then, the centroids from the results of this K-Means are used as the initial centroids for the K-Means that are performed with all input vectors. In this way, the clustering speed can be accelerated since fewer iterations are needed when the initialized centroids are close to the centroids of the final clustering result. In this work, K-Means is executed with the hierarchical sampling technique, which means that the sampling technique is repeatedly applied to different hierarchical levels. Fig. 1 illustrates this process. When the level is 3, the sample rate is $(1/2^3)$, which means that one input vector is sampled from 2^3 input vectors. Similarly, when the level is 2, the sample rate is $(1/2^2)$. The hierarchical level is reduced sequentially and is equal to 0 when all the input vectors are used.

The algorithm of K-Means clustering with the hierarchical data sampling employed in this work allows the estimation of numbers of clusters to conform to more multimedia applications. The parameters are defined as follows. K_S and K_E represent the starting number of clusters and the ending number of clusters in K-Means, respectively, and the optimal number of clusters is selected from the range of $[K_S, K_E]$ according to the BIC score [17] [18]. The higher the BIC score, the better the clustering results. The number of clusters of the clustering results with the highest BIC score is chosen as the optimal number of clusters. H denotes the total level of the hierarchical data sampling, and 2^H is less than or equal to (N/K_E), where N represents the data number of input vectors. T is the total number of trials, which is the number of times to perform K-Means with different initial centroids. The algorithm of K-Means with the hierarchical data sampling is stated as follows.

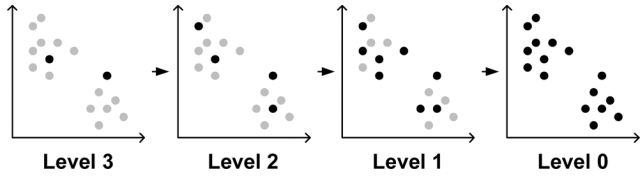

Fig. 1. Illustration of the technique of the hierarchical data sampling for K-Means. The black dots represent the data that are used for K-Means in each hierarchical level.

Step 1: Set the current level h to H, the current number of trials t to 1, the current number of clusters k to K_S, and the highest BIC score to the minimum value.

Step 2: Randomly initialize the k centroids of K-Means.

Step 3: Perform K-Means with input vectors by using a sample rate of $(1/2^h)$.

Step 4: If $h = 0$, go to **Step 5**. Otherwise, decrease h by 1 and store the centroid vectors of the clustering results as the initialized centroids for the K-Means of the next hierarchical level. Go to **Step 3**.

Step 5: Compute the maximum likelihood for the variance [17] according to the clustering results of K-Means and calculate the score of the BIC.

Step 6: If the current BIC score is higher than the highest BIC score, replace the highest BIC score with the current BIC score and store the centroids for the representation of the clustering result.

Step 7: If $t = T$, set t to 1 and go to **Step 8**. Otherwise, increase t by 1 and go to **Step 2**.

Step 8: If $k = K_E$, the algorithm is finished, and the cluster labels can be calculated according to the centroids with the highest BIC score, which corresponds to the optimal number of clusters selected from the range of $[K_S, K_E]$. Otherwise, increase k by 1 and go to **Step 2**.

Fig. 2 shows an analysis of the number of iterations in K-Means with the hierarchical sampling technique. The dark color represents the number of iterations in the hierarchical level, whereas the light color represents the accumulated numbers of iterations from previous levels. The light color in the last level (hierarchical level 0) represents the summation of numbers indicated by the dark color from previous levels (hierarchical level 1–10). In Fig. 2(a), it is shown that the numbers of iterations in most hierarchical levels are lower than the number of iterations in the initial level. This means that the algorithm of K-Means can terminate early with the hierarchical data sampling. Although the total number of iterations of all hierarchical levels, which is the accumulated number of iterations in the last hierarchical level, is higher than 40, the computations in each hierarchical level are not the same. The higher the hierarchical level, the fewer input vectors are

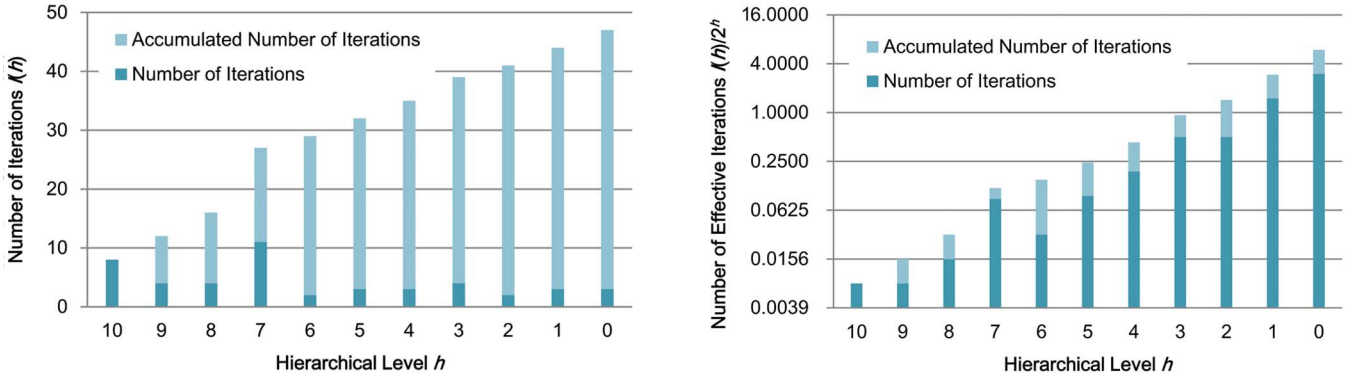

Fig. 2. (a) Number of iterations and the accumulated number of iterations based on the hierarchical data sampling. (b) Effective number of iterations and the accumulated number of effective iterations based on the hierarchical data sampling. Note that the results of the analysis are obtained by color clustering. The number of clusters in this analysis is set to $K = 4$, the number of trials is 1, and the maximum number of iterations is set to 32.

processed. Therefore, the total effective number of iterations, which can be used to compare with the number of iterations in the traditional K-Means, is defined as follows:

$$I_{\text{eff}} = \sum_{h=0}^{H} \frac{I(h)}{2^h} \tag{1}$$

where H represents the total level of the hierarchical data sampling, and $I(h)$ represents the number of iterations in the hierarchical level h. Fig. 2(b) shows the effective number of iterations accumulated in each hierarchical level, and the numbers are presented in a logarithmic scale on the ordinate axis. It is shown that the total number of effective iterations, which is the accumulated number of iterations in the last hierarchical level, is lower than 6.

Based on the definition of the effective number of iterations, in Fig. 3(a), the relation between the number of iterations and the number of clusters is analyzed. It is shown that the average number of iterations (the number of effective iterations) of the hierarchical sampling method is lower than the average number of iterations in the traditional K-Means when the number of clusters is adjusted from 2 to 16. It means that the hierarchical data sampling can effectively reduce the iterations of K-Means and save the computational time. Moreover, the relation between the total distortion and the number of clusters is shown in Fig. 3(b), which explains that the clustering qualities with the hierarchical data sampling and without the hierarchical data sampling are approximately the same. The total distortion, which is proportional to the maximum likelihood for the variance, is defined as follows:

$$\Delta = \frac{1}{N} \sum_{i=1}^{N} \left(\mathbf{x}_i - \boldsymbol{\mu}_{(i)} \right)^2 \tag{2}$$

where $\boldsymbol{\mu}_{(i)}$ denotes the centroid that is the closest to the input vector $\mathbf{x}_i$ according to the Euclidean distance measurement, and N is the total data number. The iteration of K-Means stops if the maximum number of iterations is reached or $|\Delta^{(j+1)} - \Delta^{(j)}| < \mathcal{T}_\Delta$, where j denotes the current number of iterations, and $\mathcal{T}_\Delta$ is a small positive constant. The lower the total distortion, the better the clustering quality. Although the clustering process of K-Means is accelerated by the hierarchical data sampling, the quality does not deteriorate with the reduction of computations.

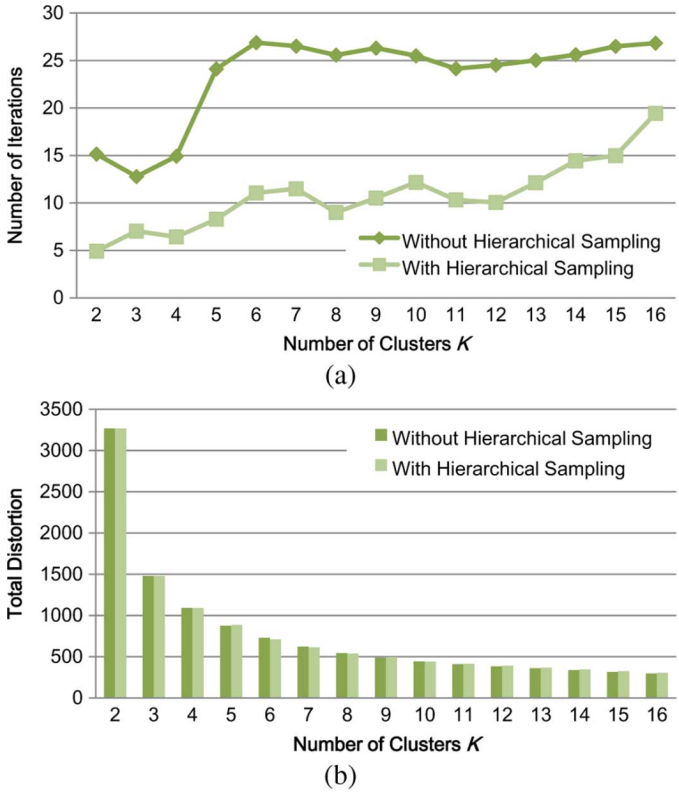

Fig. 3. (a) Relation between the number of iterations and the number of clusters based on the hierarchical sampling technique. (b) Relation between the total distortion and the number of clusters based on the hierarchical sampling technique. Note that the results of these two analyses are the average of 1000 times of experiments based on color clustering. The number of trials is 1, and the maximum number of iterations is set to 32.

III. Overview of Hardware Architecture

The proposed efficient K-Means hardware is designed to work on the system platform for multimedia processing that is shown in Fig. 4, where the CPU, the external memory, and other SIPs share the same bus resources with the K-Means architecture [15]. Connected to the system bus, the "Data Memory" is used to store the feature data that are extracted for multimedia applications, and the feature data are regarded

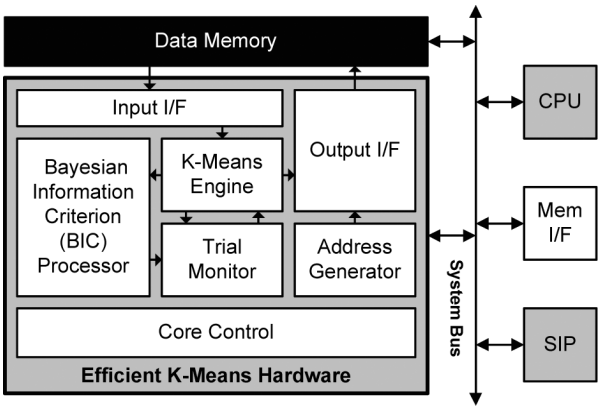

Fig. 4. Overview of the system environment and the efficient K-Means architecture, in which three main modules are included: the "K-Means Engine," the "Trial Monitor," and the "BIC Processor."

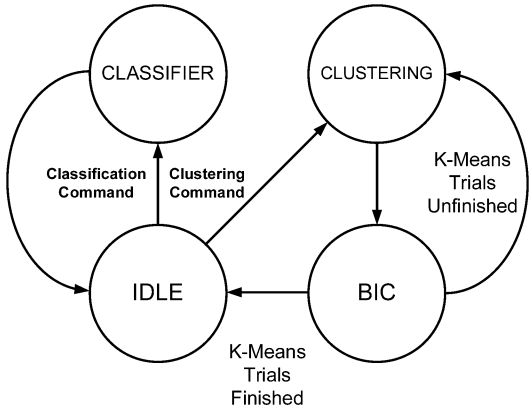

Fig. 5. Global finite-state machine of the efficient K-Means hardware.

as input vectors for K-Means clustering. The "Data Memory" can offer input vectors to the K-Means architecture with a throughput of 1 vector/cycle for iterative vector processing, so that the bandwidth resources of the system bus can be saved. The size of the "Data Memory" can be determined according to the parameters of the target application, and it functions as a local memory for the proposed hardware. When the "Data Memory" contains all input vectors used for the K-Means algorithm, the clustering results can be optimized successively without accessing the system bus.

The proposed design methodology can be implemented with different specifications. In this work, the specifications are defined based on the system platform with the target applications, which focus on low design costs and energy efficiency. The maximum number of vector dimensions D is set to 4, the bit length of each dimension B is set to 8, and the maximum number of clusters K is set to 4. The 4-D input vectors can be sent to the hardware in one cycle through the 32-bit system bus. As shown in Fig. 4, the K-Means hardware consists essentially of three main modules. The first module, the "Bayesian-Information-Criterion (BIC) Processor" is used to compute the BIC scores of the clustering results of K-Means based on logarithm operations to estimate numbers of clusters, and the number of clusters associated with the highest BIC score is selected. The second module, the "Trial Monitor" preserves the best clustering results of K-Means trials by storing the corresponding cluster centroids. The last module, the "K-Means Engine" performs the iterative clustering process of K-Means with the hierarchical sampling technique and communicates with the two modules mentioned above. In this section, the functionalities of these modules and state transitions are introduced. The architecture of the "BIC Processor," which is one of the important contributions of this work, is described in Section IV.

Contained in the "Core Control" shown in Fig. 4, the global finite-state machine of the efficient K-Means hardware is shown in Fig. 5, which includes four states. The state transitions are explained with the interactions among the modules of the K-Means hardware. Initially, the state is IDLE, and two commands for clustering and classification can be issued externally. After receiving the clustering command, K-Means clustering is performed in the CLUSTERING state in the "K-Means Engine." The requests and addresses for the input vectors with the hierarchical sampling patterns are generated by the "Address Generator," which is connected to the "Output Interface" in Fig. 4. After a single trial of K-Means is finished, the clustering results are sent to the "BIC Processor" to compute the BIC score, which is then stored in the "Trial Monitor" with the cluster centroids from the "K-Means Engine." If the K-Means trials are not finished, K-Means clustering is performed again with different centroid initializations. In the BIC state, the BIC score of this trial of K-Means is also computed, and the cluster centroids in the "Trial Monitor" are updated if the BIC score is higher than the one stored in the "Trial Monitor." Based on this procedure, the clustering results with the highest BIC score can be preserved. Not only can the number of clusters be determined, the clustering results with higher qualities than existing K-Means architectures can also be generated. When all the trials of K-Means are finished, the operations stop and the state returns to IDLE. Then, the labels of input vectors can be retrieved by issuing the classification command, and the clustering results with the best BIC score stored in the "Trial Monitor" are dumped out in the CLASSIFIER state. The state transitions of the global finite-state machine are consistent with the algorithm mentioned in Section II.

The "K-Means Engine" contains four "EM" modules [15], which are also called the "E-M Distance Calculators" [13], to handle 4-D vectors and compute the Euclidean distances and the Manhattan distances by using the same hardware resources. The data gating mechanism is employed in distance calculations to save power consumption for different numbers of clusters. The "Vector Dividers" [15] are also included to compute the divisions for the variance and the updated centroids. In order to handle the variance computation, the summation of the Euclidean distances has to be accumulated, and the bit length of the dividend and the divisor needs to be extended from the bit length of the ones used in our previous works [13], [15]. The local finite-state machine, which is shown in Fig. 6, is contained in the "K-Means Engine." It corresponds to the CLUSTERING state of the global finite-state machine and consists of the states to handle the data sampling in each hierarchical level. In the INITIAL state, the "K-Means Engine" sends the request for the vectors that are used for the centroid initialization. Next, in the CHECK state, whether the

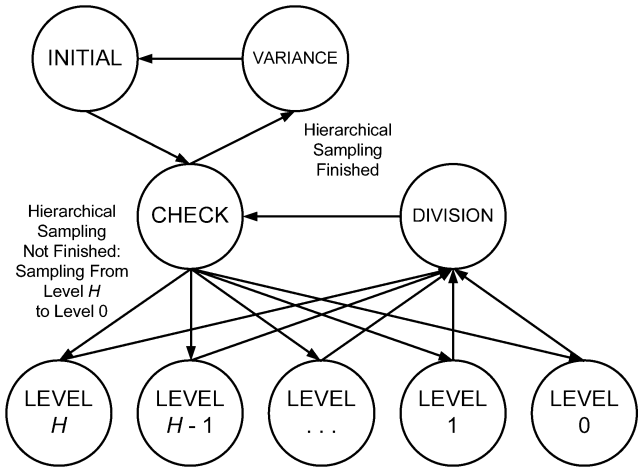

Fig. 6. Local finite-state machine in the CLUSTERING state of the global finite-state machine of the efficient K-Means hardware.

iterations based on the hierarchical data sampling are finished or not is checked. The hierarchical data sampling starts from the highest level to the lowest level. The iterative process of K-Means continues in the current sampling level if the iterations are not finished. Otherwise, the sampling level of K-Means proceeds to the next level. Each level of the sampling corresponds to one state, and the divisions for the cluster centroid updating are performed in the DIVISION state after one iteration in each level of the hierarchical data sampling is over. When the maximum number of iterations is reached or the K-Means iterative procedure terminates, the centroids are computed in the DIVISION state and regarded as the initial centroids in the following level. After the LEVEL 0 state is reached and K-Means iterations are finished, the centroids that represent the clustering results are generated in the DIVISION state. The variance computation for the BIC score is completed in the VARIANCE state with the "Vector Dividers" based on the accumulated Euclidean distances and the data number of input vectors.

IV. Bayesian-Information-Criterion Processor

The BIC, also known as the Schwarz criterion, is a method that can be employed to estimate the suitable number of clusters K. The BIC formula is defined as follows [17], [18]:

$$\mathrm{BIC}(M_j) = \hat{l}_j(\delta) - \frac{p_j}{2}\log N \tag{3}$$

where $\hat{l}_j(\delta)$ is the log likelihood of the data δ according to the jth model and taken at the maximum likelihood point, and p_j is the number of parameters in M_j, which is equal to $(D+1)\times K$ under the identical spherical Gaussian distribution [16]–[18]. Note that D denotes the number of dimensions of each input vector. The model with the largest score is selected, and the number of clusters can be estimated. The maximum likelihood estimate for the variance is shown as follows:

$$\hat{\sigma}^2 = \frac{1}{N-K}\sum_{i=1}^{N}\left(\mathbf{x}_i - \boldsymbol{\mu}_{(i)}\right)^2 \tag{4}$$

where (i) denotes the index of the centroid that is the closest to the input vector $\mathbf{x}_i$, N is the data number of input vectors, and $\boldsymbol{\mu}_{(i)}$ is the centroid associated with the input vector $\mathbf{x}_i$. It can be computed from the total distortion Δ defined in (2). Based on the point probabilities [17], [18], the log-likelihood of the data can be calculated as follows:

$$\begin{aligned}\hat{l}(\delta) &= \log\prod_{i=1}^{N}\hat{P}(\mathbf{x}_i)\\ &= \sum_{i=1}^{N}\left(\log\frac{1}{\sqrt{2\pi}\hat{\sigma}^D} - \frac{1}{2\hat{\sigma}^2}\left|\mathbf{x}_i - \boldsymbol{\mu}_{(i)}\right|^2 + \log\frac{N_i}{N}\right)\\ &= -\frac{N}{2}\log 2\pi - \frac{ND}{2}\log\hat{\sigma}^2 - \frac{N-K}{2}\\ &\quad + \sum_{i=1}^{K}N_i\log N_i - N\log N.\end{aligned} \tag{5}$$

Since the purpose of computing BIC scores is to estimate the number of clusters of the same set of input vectors, (5) can be simplified. The first term, the fifth term, and part of the third term do not change with the same set of input vectors, so they can be ignored when BIC scores are compared. By multiplying (5) by 2 and reducing the first term, the fifth term, and parts of the third term, the modified BIC formula can be obtained as follows:

$$\begin{aligned}&\widehat{\mathrm{BIC}}(M_j)\\ &= -ND\log\hat{\sigma}^2 + K + 2\sum_{i=1}^{K}N_i\log N_i - p_j\log N\\ &= \sum_{i=1}^{K}N_i\left(2\log N_i - D\log\hat{\sigma}^2\right) - p_j\log N + K\end{aligned} \tag{6}$$

which gives the same comparison results as the BIC score computed based on (5) when the same set of input vectors are considered. Then, to reduce the hardware costs and the design complexity, two schemes are applied here. The first is the reuse of the same multiplier. Due to the high hardware costs for long bit-length multipliers, only one multiplier is employed in the "BIC Processor." Besides, to reduce the design costs of multiplications with negative numbers, constants α and β are added to the equation to make all the input values positive before performing multiplications. Since BIC scores are used for comparisons, adding constants to (6) does not affect the final result. The hardware-oriented BIC formula is shown as follows:

$$\begin{aligned}\widetilde{\mathrm{BIC}}(M_j) = \sum_{i=1}^{K}N_i\left[2\log N_i + (\alpha - D\log\hat{\sigma}^2)\right]\\ +(\beta - p_j\log N) + K\end{aligned} \tag{7}$$

which contains additions, subtractions, multiplications, and logarithm operations. The precision analysis and the hardware architectures are introduced in the following subsections.

A. Precision Analysis and Architecture of Log Processor

To compute the score of BIC based on (7), the computations of the natural logarithm function are inevitable. The analysis and

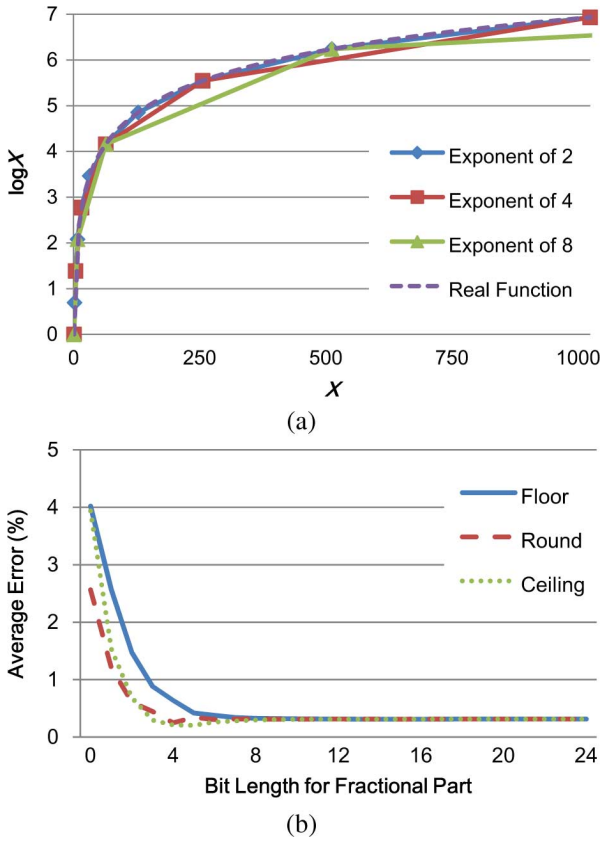

Fig. 7. (a) Natural logarithm function and approximations with powers of different integers. (b) Relation between the precision error and the bit length for the fractional part with different methods.

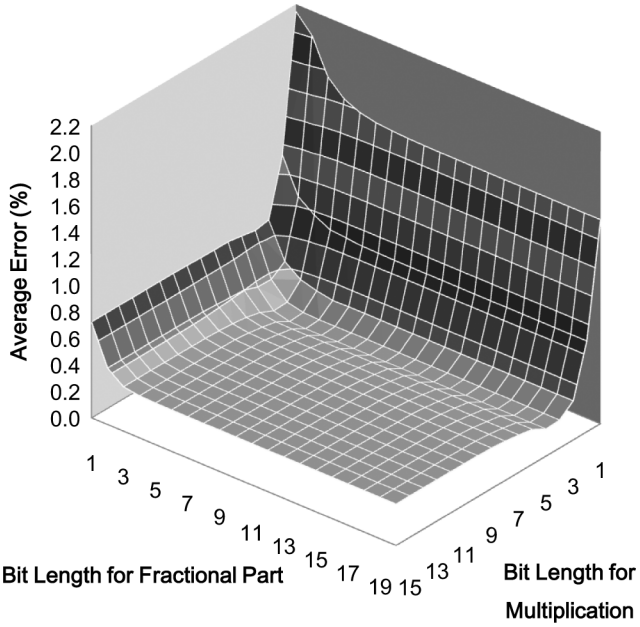

Fig. 8. Relation among the precision error, the bit length for the multiplication, and the bit length for the fractional part.

design of the "Log Processor" are introduced in this subsection. Since the maximum data number of the K-Means hardware is set to 2^{20}, the maximum input value of the logarithm function is also equal to this value. Based on the cost analysis, it is not possible to store 2^{20} entries for the logarithm look-up table (LUT). Therefore, the LUT based on the linear interpolation is adopted. To match up the value variation of the logarithm function, the powers of integers are used for sample points of interpolation. Fig. 7(a) shows three examples of the natural logarithm function based on the linear interpolation by using sample points with powers of 2, 4, and 8. It is clearly shown that using powers of 2 can approximate the logarithm function the best. In the design of the "Log Processor" architecture, a total of 21 sample points, $\log(2^0), \ldots, \log(2^{20})$, are stored in the LUT, and the approximated logarithm function, which is suitable for hardware implementation, is expressed in the following formula:

$$\widetilde{\log}(x) = \log\left(2^{\lfloor \log_2 x \rfloor}\right) + \left(x - 2^{\lfloor \log_2 x \rfloor}\right) \frac{\log 2}{2^{\lfloor \log_2 x \rfloor}}. \quad (8)$$

In order to achieve low precision errors in the hardware architecture, the relation between the precision error and the bit length for the fractional part with different methods is analyzed in Fig. 7(b). The result shows that the precision errors based on the floor function, the round function, and the ceiling function are approximately the same when the bit length of the fractional part is long. However, when the bit length of the fractional part is short, the precision errors of the floor function, the round function, and the ceiling function become different. The ceiling function gives the best result when the bit length of the fractional part is equal to 5. Based on the ceiling function, the bit length for the multiplication in the linear interpolation is also analyzed, and the relation among the precision error, the bit length for the multiplication, and the bit length for the fractional part, is shown in Fig. 8 as a 3-D plot. It is observed that the variation of the error is stable as the parameters become large, and the increment of the bit lengths does not necessarily reduce the error. By considering the trade-off of these parameters and the error, the local minimum of the precision can be found when the bit length for the multiplication is 5 and the bit length for the fractional part is 5. Note that the bit lengths are only considered in internal processing of the "Log Processor."

Based on the analysis above, the architecture of the "Log Processor" is established in Fig. 9. There are four pipeline stages in this architecture, and the format of input vectors is the fixed-point integer. In the first stage, the "Most-Significant-Bit (MSB) Locator" is employed to find the position of the MSB of the input number, and the output of the "MSB Locator" includes the position of the MSB and the lower sample point that is the closest to the input value. Note that the sample points are all expressed as powers of 2. Then, in the second stage, the difference of input number and the lower sample point is obtained, and two logarithm values are read from the "Dual Log LUT" to compute the difference, which is the value of $\log 2$ in the numerator of (8). These two logarithm values correspond to the lower and the upper sample points that are the closest to the input, respectively, and they are represented as the fixed-point format with the integer part and the fractional part. Next, in the third stage, the product of the input difference and the logarithm difference is obtained for the interpolation. Finally, in the fourth stage, the division operation is saved by ridding of redundant

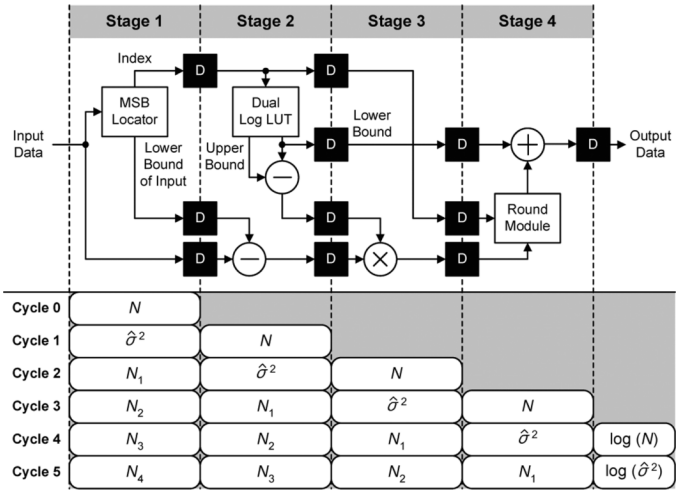

Fig. 9. Architecture of the "Log Processor" and the detailed pipeline stages.

bits with the "Round Module," and the interpolated logarithm value which corresponds to the input number is generated with both the integer part and the fractional part.

B. Architecture of BIC Processor

The architecture of the "BIC Processor," which is shown in Fig. 10, is designed based on (7). It contains the "Log Processor," two processing element (PE) sets (the "BIC PE Set α" and the "BIC PE Set β"), and an accumulator. The function of the "Log Processor," whose architecture is described in Section IV, is to compute the natural logarithm function with a throughput of 1 output/cycle. After receiving the output value from the "Log Processor," The "BIC PE Set β" is employed to compute the value of $(\beta - p_j \log N)$ in (7). In the "BIC PE Set α," a three-stage multiplier that can process 20-bit $\times$ 12-bit multiplications is included to compute the value of $N_i[2 \log N_i + (\alpha - D \log \hat{\sigma}^2)]$ in (7) for different clusters. By combining the outputs of the two PE sets, the result of (7) can be obtained. The five pipeline stages of the latter half of the "BIC Processor" are also shown in Fig. 10. It takes two pipeline stages and four pipeline stages to complete the computations in the "BIC PE Set β" and the "BIC PE Set α," respectively. In the fifth pipeline stage, the BIC score is summed up. Altogether $K + 5$ clock cycles are needed in the latter half of the "BIC Processor" to compute the BIC scores. Therefore, 9 cycles are required when the number of clusters $K = 4$. Based on the hardware architecture of the "BIC Processor," the BIC scores for different numbers of clusters K can be computed and employed to select the optimal number of clusters.

V. Experimental Results

The experiments, which contain five parts, are discussed in the following subsections. First, the algorithm of the proposed method is verified and the applications with the efficient K-Means hardware are shown. Then, the architectural analysis of the hardware is performed. Next, the comparisons of hardware specifications with previous works are discussed, and the performance of the proposed algorithm is evaluated on a GPU to be compared with the efficient K-Means architecture. Last

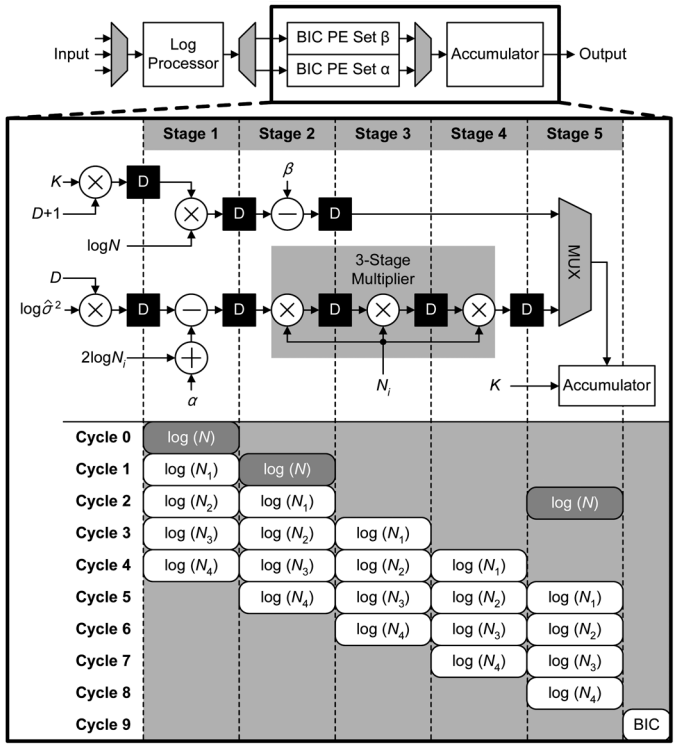

Fig. 10. Architecture of the "BIC processor" and the detailed pipeline stages.

but not least, the power scalability of the efficient K-Means hardware is described.

A. Algorithm Verification and Applications

The multimedia applications including image segmentation [8], [13] and color clustering, which are implemented in our previous works [13], [15], are tested using the proposed hardware. However, different from existing works, two new applications are demonstrated in this subsection. The first is motion segmentation, and the second is edge-adaptive noise reduction. K-Means can be applied to motion segmentation for moving image representation [20]. In this implementation, each input vector has three dimensions, including two dimensions for motion vectors and one dimension for motion magnitudes. Image blocks with similar motion directions are prone to be in the same cluster. For edge-adaptive noise reduction, K-Means clustering is performed with the pixels in each window of the input image as a filtering process. The number of clusters is estimated based on the BIC score, and the color of the cluster centroid that the center pixel in the window belongs to is used to represent the filtering result of each window.

The results of both motion segmentation and edge-adaptive noise reduction obtained from the software simulation are exactly the same as the ones obtained from the hardware computation based on Verilog HDL simulation. Fig. 11 shows some motion segmentation results of different frames in two CIF-size video sequences, and the most suitable number of clusters is chosen for each frame. In one sequence, Fig. 11(b) is the segmentation result of Fig. 11(a), and Fig. 11(c) shows the motion vectors, which are represented as black lines and generated based on overlapping 16×16-pixel windows with a 2-D

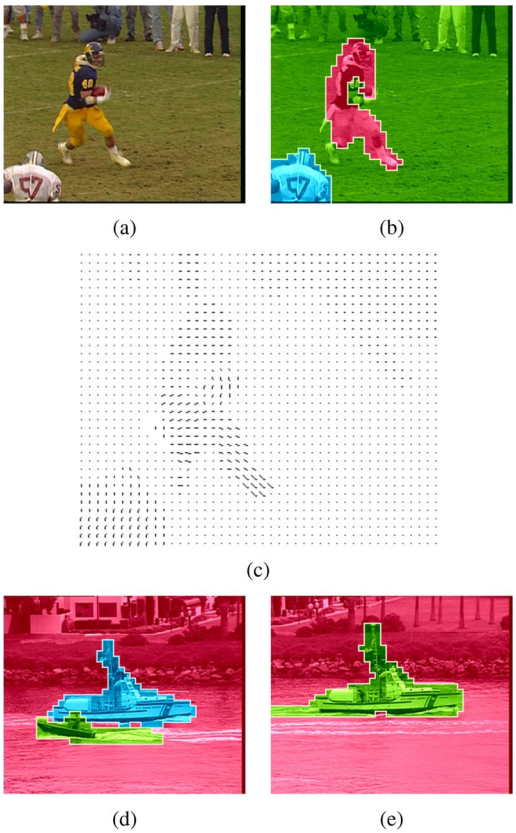

Fig. 11. Example of motion segmentation based on the proposed hardware architecture. (a) Frame 126 of the video sequence, "Football." (b) The motion segmentation result of frame 126 in "Football." (c) The motion vectors of frame 126 of the video sequence, "Football." (d) The motion segmentation result of frame 85 in "Coastguard." (e) The motion segmentation result of frame 194 in "Coastguard."

search range of $(\pm 16, \pm 16)$ in pixel. In the other sequence, the estimated numbers of clusters are $K = 3$ and $K = 2$ in Fig. 11(d) and (e), respectively. The segmentation results can be applied to frame-rate up-conversion (FRUC) [21] or video content analysis. Fig. 12 shows an example of the edge-adaptive filtering by using an image with a size of 256×256 pixels. The noise reduction based on the traditional low-pass filter is shown in Fig. 12(b), where the noise is eliminated, but the high-frequency details are also blurred. The result of the edge-adaptive filtering based on K-Means with estimation of numbers of clusters is shown in Fig. 12(c), where the noise is eliminated, and the high-frequency details can be preserved. The estimated numbers of clusters in all windows are represented in different colors in Fig. 12(d). Based on the BIC computations, it can be observed that the number of clusters becomes larger in the windows that cover image edges. It is reasonable since edges usually appear in a window when there are more than two color clusters. It takes more than 10 s for a Pentium III 3.2 GHz CPU with 2 GB SDRAM to complete the filtering based on K-Means, but the proposed hardware only requires less than 2 s under the condition that the operating frequency is 233 MHz and the bandwidth of the system bus is available.

B. Architectural Analysis

In this part, the critical path and hardware costs of each module in the efficient K-Means hardware architecture are analyzed using the logic synthesis tool, Synopsys Design Vision, with the TSMC 90 nm technology library. The maximum clock frequency achieves 625 MHz, the total gate count is 51 K, and the corresponding area is 0.14 mm^2. The high frequency results from the pipeline techniques employed in the hardware design. The area breakdown of each module is shown in Fig. 13, where the "K-Means Engine" occupies 76% of the whole design. The "BIC Processor," which is the key module to estimate numbers of clusters, occupies 13% of the area, which is equivalent to 0.02 mm^2. The results reflect the low-cost property of the architecture, and the small area reduces the overhead to integrate the efficient K-Means hardware with other complicated multimedia systems.

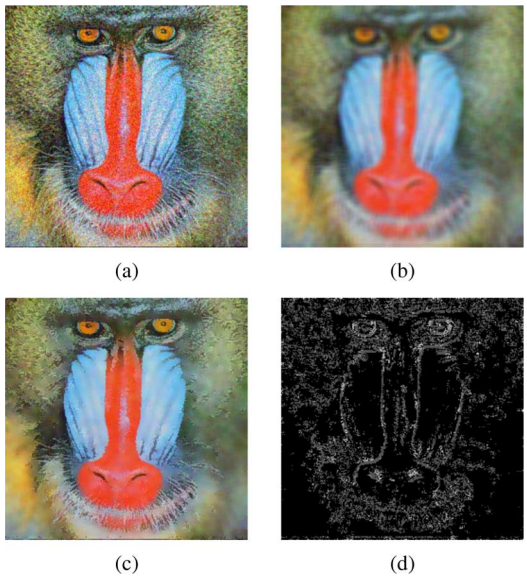

Fig. 12. Example of edge-adaptive noise reduction based on the proposed hardware architecture. (a) The original image, "Baboon," which is corrupted with Gaussian noise ($\mathrm{PSNR} = 19.78$ dB). (b) The noise reduction result based on the traditional low-pass filter ($\mathrm{PSNR} = 21.28$ dB). (c) The noise reduction result based on the efficient K-Means hardware ($\mathrm{PSNR} = 22.89$ dB). (d) The result of estimation of numbers of clusters based on the efficient K-Means hardware. Note that there are four gray-scale levels, which represent numbers of clusters from $K = 1$ to $K = 4$. The lower the gray-scale level, the smaller the number of clusters.

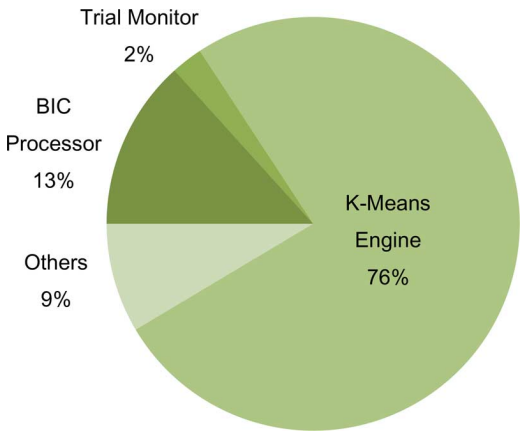

Fig. 13. Area percentage breakdown of main modules in the efficient K-Means hardware.

C. Comparison of Hardware Specifications

The comparison results of hardware specifications are summarized in Table I, where the proposed work is compared with previous works [12], [13], [15]. Note that the gate counts do not include the local memory for input vectors. This work achieves the highest clock frequency among previous works by using the 90-nm complimentary metal-oxide semiconductor (CMOS) technology. Although this work supports smaller numbers of clusters and lower-dimensional vectors than previous works, the functionality is enough for the testing applications. Besides, the area of this work is the smallest among all the works. By computing the gate count per maximum throughput for all of our works, the results show that this work achieves the lowest value, 13 K cycle/dimension, which conforms to the low-cost goal for the K-Means architecture. Moreover, this work supports the hierarchical sampling technique for speed acceleration, which can make the clustering process terminate earlier than the traditional K-Means. Furthermore, the clustering quality of this work can be improved by increasing the total number of trials, which can not be supported by previous works. The architecture to estimate numbers of clusters, which is not included in previous works and related works [6]–[11], is also integrated into the efficient K-Means hardware. By combining the low-cost advantage, the hierarchical sampling technique, and the functionality to estimate numbers of clusters, the proposed K-Means architecture can be employed efficiently for applications that are different from existing works.

D. GPU Implementation and Performance Evaluation

While evaluating the efficient K-Means hardware architecture, the modern GPU which contains highly-parallel stream processing architectures is also compared. Recently, modern GPU architectures have become more and more flexible. Not only the high-quality rendering algorithms are supported, general-purpose computations are also mapped to the graphics hardware. The technique is called general-purpose computations on graphics processing units (GPGPU) [23]. Sun's GPU [24], [25], which supports OpenGL ES 2.0 and OpenGL Shading Language, is adopted for the implementation of K-Means. The GPU model is improved from the architecture of the previous chip design [26]. Since the GPU is also designed for embedded systems of mobile devices, it is suitable for the comparison with the proposed hardware. The details of the implementation are given in the Appendix.

Table II shows the specifications of the GPU [25] with the K-Means implementation. The throughput of the GPU is close to the efficient K-Means hardware by using 16 single-instruction multiple-data (SIMD) cores, and both the maximum data number and the maximum number of clusters are higher than the K-Means architecture. The performance of the algorithm is proportional to the number of SIMD cores in the GPU. Since the computation power of 16 cores is equal to typical commercial off-the-shelf (COTS) mobile graphics chips, the 16-SIMD-core implementation is chosen for comparison. The GPU shows its ability to support K-Means clustering with high specifications. However, the main overhead of supporting these features for K-Means clustering is the gate count, which is 52.9 times higher than the K-Means hardware. Besides, the gate count per throughput of the GPU is approximately 63 times higher than the K-Means hardware. Clearly, this comparison manifests the low-cost advantages of the proposed K-Means hardware architecture.

E. Power Consumption

To compare the power consumption of previous works and this work, a K-Means accelerator with 16-parallel "EM" modules [15] that can process 64 clusters is developed based on the hardware architecture in our previous works [12], [13]. The accelerator is implemented with the TSMC 90 nm technology, and its processing capability is higher than the efficient K-Means architecture. The maximum throughput of this accelerator is 5 dimension/cycle, and the power consumption of the accelerator is 209 mW when the clock frequency is 500 MHz and $K = 64$.

In this work, the efficient K-Means architecture is implemented with the TSMC 90 nm technology by using the place-and-route tool, Cadence SoC Encounter to analyze the power dissipation. The core size is 0.7 mm $\times$ 0.7 mm, and a part of the core area is reserved for other designs. The result is shown in Fig. 14, where the power consumption is analyzed in different vector dimensions and different values of K. The power consumption of the hardware achieves 12.95 mW with 4-D vectors and four clusters when the operating frequency is 233 MHz. In the case of processing 1-D vectors with only one cluster, the power consumption is 10.72 mW, and 17.2% of power is saved. It is shown that the power consumption of the hardware is approximately proportional to the number of clusters K and the number of dimensions, and the power scalability results from the data gating mechanism for distance calculations. Because of the low power consumption, the efficient architecture is applicable to systems in energy-aware devices.

VI. Conclusion and Future Work

A power-efficient and low-cost K-Means hardware architecture based on the hierarchical sampling technique is proposed with the "BIC Processor" to estimate numbers of clusters. The integration of the hierarchical data sampling in the hardware accelerates the clustering speed of K-Means to achieve higher efficiency than existing works. The proposed architecture can be used for motion segmentation and edge-adaptive noise reduction with low dimensions and small numbers of clusters, and the strengths of the hardware are verified by comparing with the GPU in the experiments.

This work focuses on low area costs and low power dissipation of K-Means to provide new strategies for multimedia processing in the resource-limited embedded systems, and the power consumption scales well with the number of clusters and the number of dimensions. The design mechanism of the "BIC Processor" can also be extended to other K-Means architectures with different specifications. For future developments, a robust K-Means hardware architecture that can handle evolutionary clustering problems might be the next target.

Appendix

The data flow for K-Means clustering with the hierarchical sampling technique in the GPGPU implementation with Sun's GPU [24], [25] is shown in Fig. 15, where the single iteration

TABLE I
COMPARISON OF THIS WORK AND PREVIOUS WORKS

	K-Means SIP (ISCAS 2008 [12])	Bandwidth Adaptive K-Means (TVLSI 2010 [13])	Hierarchical K-Means (TVLSI 2011 [15])	Efficient K-Means (This Work)
Technology	TSMC 0.18μm	TSMC 90nm	TSMC 90nm	TSMC 90nm
Clock Frequency	200MHz	400MHz	333MHz	625MHz
Number of K	1 – 16	1 – 16	2 – 1024	1 – 4
Gate Count	58K	440K	414K	51K
Area	0.58mm^2	1.23mm^2	1.16mm^2	0.14mm^2
Normalized Area*	0.15mm^2	1.23mm^2	1.16mm^2	0.14mm^2
Vector Dimension	1 – 5	1 – 16	1 – 8	1 – 4
Maximum Throughput	2.5 Dimension/Cycle	16 Dimension/Cycle	8 Dimension/Cycle	4 Dimension/Cycle
Gate Count per Throughput**	23K Cycle/Dimension	28K Cycle/Dimension	52K Cycle/Dimension	13K Cycle/Dimension
Initialization Method	Predefined Centroid	Forgy [22]	Forgy [22]	Forgy [22]
Distance Measurement	Manhattan	Euclidean/Manhattan	Euclidean/Manhattan	Euclidean/Manhattan
Maximum Data Number	2^{17}	2^{20}	2^{24}	2^{20}
Highlights	Prototype	1. Highest Dimension 2. Bandwidth Adaptive Mechanism	1. Largest K 2. Hierarchical Clustering	1. Lowest Gate Count 2. Estimation of K

* The normalized area refers to the area normalized to 90nm, and the normalization factor from 0.18μm to 90nm is $(\frac{90}{180})^2$.
** The gate count per throughput refers to the gate count normalized with the maximum throughput.

TABLE II
SPECIFICATIONS OF GPU IMPLEMENTATION

	Sun's GPU (COOLChips XIII [25])
Technology	UMC 90nm
Clock Frequency	200MHz
Number of K	1 – 1024
Gate Count	2.7M (16 SIMD Cores)
Area	30mm^2
Vector Dimension	1 – 4
Maximum Throughput	3.28 Dimension/Cycle
Gate Count per Throughput*	823K Cycle/Dimension
Initialization Method	Forgy [22]
Distance Measurement	Euclidean/Manhattan
Maximum Data Number	2^{32}

* The gate count per throughput refers to the gate count normalized with the maximum throughput.

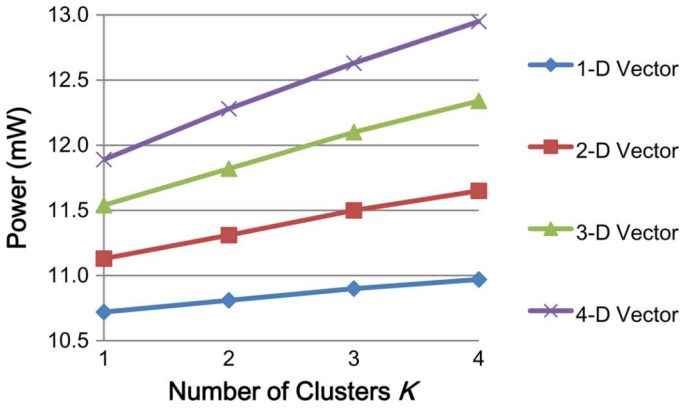

Fig. 14. Power consumption of the efficient K-Means hardware when the operating frequency is 233 MHz.

of K-Means clustering can be divided into two stages, which are the Nearest Centroid Computation stage and the Centroid Update stage.

In the *Nearest Centroid Computation* stage, each polygon represents one sampling data of the current iteration. The width of each polygon is equal to the total number of trials, and the

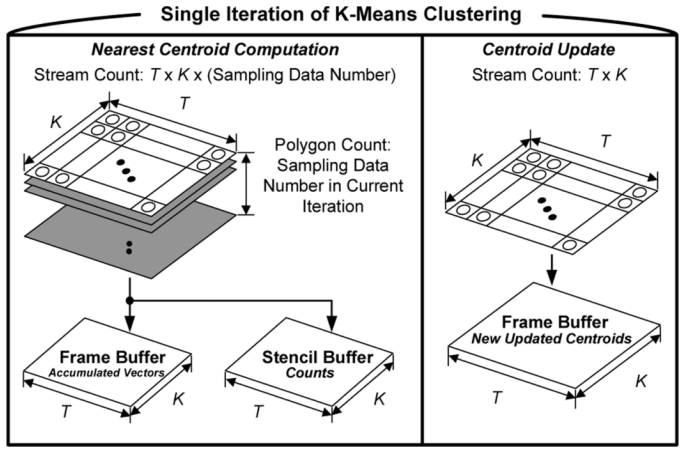

Fig. 15. Illustration of the data flow of K-Means with the hierarchical data sampling in the GPU. T and K denote the total number of trials and the number of clusters, respectively.

height is equal to the number of clusters. Each point in the polygon represents one cluster in one trial. In the same column (one of the trials), when the distance between the current polygon (one of the sampling data) and the point (one of the clusters) is the minimum, the output vectors stored in the point are rendered for the current sampling data. Otherwise, the point is discarded. The blending function and the stencil function in the adding mode are set to accumulate the output vectors of each point and the number of rendered points into the frame buffer and the stencil buffer, respectively. In other words, after all polygons are processed, the results in the frame buffer and the stencil buffer represent the accumulated vectors and the count for each cluster, respectively.

In the *Centroid Update* stage, there is only one polygon that needs be processed. The only computation of each point is the division operation, where the corresponding values in the frame buffer (accumulated vectors) are divided by the corresponding value in the stencil buffer (count). Afterwards, the new centroids are generated in the current frame buffer to be used for the next iteration.

ACKNOWLEDGMENT

The authors thank National Chip Implementation Center (CIC), Taiwan, for EDA tool support and TSMC University Shuttle Program for process support.

REFERENCES

[1] T.-D. Chuang, P.-K. Tsung, P.-C. Lin, L.-M. Chang, T.-C. Ma, Y.-H. Chen, and L.-G. Chen, "A 59.5 mW scalable/multi-view video decoder chip for quad/3D full HDTV and video streaming applications," in *Dig. Tech. Papers IEEE Int. Solid-State Circuits Conf.*, Feb. 2010, pp. 330–331.

[2] T. Kurafuji, M. Haraguchi, M. Nakajima, T. Gyoten, T. Nishijima, H. Yamasaki, Y. Imai, M. Ishizaki, T. Kumaki, Y. Okuno, T. Koide, H. J. Mattausch, and K. Arimoto, "A scalable massively parallel processor for real-time image processing," in *Dig. Tech. Papers IEEE Int. Solid-State Circuits Conf.*, Feb. 2010, pp. 334–335.

[3] E. Alpaydin, *Introduction to Machine Learning*, 2nd ed. Cambridge, MA: MIT Press, Feb. 2010.

[4] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Stat. Probabil.*, 1967, pp. 281–297.

[5] K. Krishna and M. Narasimha Murty, "Genetic K-means algorithm," *IEEE Trans. Syst., Man, Cybern.—Part B: Cybern.*, vol. 29, no. 3, pp. 433–439, Jun. 1999.

[6] M. Estlick, M. Leeser, J. Theiler, and J. J. Szymanski, "Algorithmic transformations in the implementation of K-means clustering on reconfigurable hardware," in *Proc. ACM/SIGDA Int. Symp. Field Programmable Gate Arrays*, Feb. 2001, pp. 103–110.

[7] A. G. da, S. Filho, A. C. Frery, C. C. de Araújo, H. Alice, J. Cerqueira, J. A. Loureiro, M. E. de Lima, M. das, G. S. Oliveira, and M. M. Horta, "Hyperspectral images clustering on reconfigurable hardware using the K-means algorithm," in *Proc. Symp. Integrated Circuits Syst. Design*, Sep. 2003, pp. 99–104.

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

[9] T. Saegusa and T. Maruyama, "An FPGA implementation of real-time K-means clustering for color images," *J. Real-Time Image Process.*, vol. 2, no. 4, pp. 309–318, Dec. 2007.

[10] Y. Ma and T. Shibata, "A real-time K-means learning processor architecture extendible to multiple-chip large-scale systems," in *Proc. 34th Eur. Solid-State Circuits Conf.*, Sep. 2008.

[11] Y. Ma and T. Shibata, "A binary-tree hierarchical multiple-chip architecture for real-time large-scale learning processor systems," *Jpn. J. Appl. Phys.*, vol. 49, no. 4, p. 04DE08, Apr. 2010.

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

[13] T.-W. Chen and S.-Y. Chien, "Bandwidth adaptive hardware architecture of K-means clustering for video analysis," *IEEE Trans. Very Large Scale Integrat. (VLSI) Syst.*, vol. 18, no. 6, pp. 957–966, Jun. 2010.

[14] T.-W. Chen, Y.-L. Chen, and S.-Y. Chien, "Photo retrieval based on spatial layout with hardware acceleration," in *Proc. IEEE Int. Symp. Intell. Signal Process. Commun. Syst.*, Dec. 2009, pp. 196–199.

[15] T.-W. Chen and S.-Y. Chien, "Flexible hardware architecture of hierarchical K-means clustering for large cluster number," *IEEE Trans. Very Large Scale Integrat. (VLSI) Syst.*, vol. 19, no. 8, pp. 1336–1345, Aug. 2011.

[16] G. Hamerly and C. Elkan, "Learning the K in K-means," in *Proc. 17th Annu. Conf. Neural Informat. Process. Syst.*, Dec. 2003, pp. 281–288.

[17] D. Pelleg and A. Moore, "X-means: Extending K-means with efficient estimation of the number of clusters," in *Proc. 17th Int. Conf. Mach. Learn.*, Jun. 2000, pp. 727–734.

[18] G. Gan, C. Ma, and J. Wu, *Data Clustering: Theory, Algorithms, and Applications*. Philadelphia, PA: SIAM, May 2007.

[19] S. Phillips, "Reducing the computation time of the Isodata and K-means unsupervised classification algorithms," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, Jun. 2002, vol. 3, pp. 1627–1629.

[20] J. Y. A. Wang and E. H. Adelson, "Representing moving images with layers," *IEEE Trans. Image Process.*, vol. 3, no. 5, pp. 625–638, Sep. 1994.

[21] B.-D. Choi, J.-W. Han, C.-S. Kim, and S.-J. Ko, "Frame rate up-conversion using perspective transform," *IEEE Trans. Consumer Electron.*, vol. 52, no. 3, pp. 975–982, Aug. 2006.

[22] J. M. Peña, J. A. Lozano, and P. Larrañaga, "An empirical comparison of four initialization methods for the K-means algorithm," *Pattern Recognit. Lett.*, vol. 20, no. 10, pp. 1027–1040, Oct. 1999.

[23] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell, "A survey of general-purpose computation on graphics hardware," *Comput. Graphics Forum*, vol. 26, no. 1, pp. 80–113, Mar. 2007.

[24] C.-H. Sun, K.-H. Lok, Y.-M. Tsao, C.-M. Chang, and S.-Y. Chien, "CFU: Multi-purpose configurable filtering unit for mobile multimedia applications on graphics hardware," in *Proc. Conf. High Performance Graphics*, Aug. 2009, pp. 29–36.

[25] C.-H. Sun, Y.-M. Tsao, K.-H. Lok, and S.-Y. Chien, "Multimedia system-on-a-chip: Low power multi-purpose GPU with multi-core stream processing unit, universal rasterizer, and mipmapping texture compression," in *Proc. IEEE Symp. Low-Power High-Speed Chips (COOL Chips XIII)*, Apr. 2010, pp. 365–367.

[26] Y.-M. Tsao, C.-H. Sun, Y.-C. Lin, K.-H. Lok, C.-J. Hsu, S.-Y. Chien, and L.-G. Chen, "A 26 mW 6.4 GFLOPS multi-core stream processor for mobile multimedia applications," in *Dig. Tech. Papers Symp. VLSI Circuits*, Jun. 2008, pp. 24–25.

Tse-Wei Chen (S'07–M'11) received the B.S. degree from the Department of Electrical Engineering, and the Ph.D. degree from the Graduate Institute of Electronics Engineering (GIEE), National Taiwan University, Taipei, Taiwan, in 2006 and 2010, respectively.

In 2010, he became a Foreign Joint Researcher with the Graduate School of Information Science, Nagoya University, Nagoya, Japan. His research interests include computer vision, pattern recognition, machine learning, and associated VLSI architectures.

Chih-Hao Sun received the B.S. degree from the Department of Electrical Engineering and the M.S. degree from the Graduate Institute of Electronics Engineering (GIEE), National Taiwan University (NTU), Taipei, Taiwan, in 2006 and 2008, respectively.

His research interests are 3-D computer graphics hardware architecture design.

Hsiao-Hang Su received the B.S. degree, in 2010, from the Department of Electrical Engineering, National Taiwan University (NTU), Taipei, Taiwan, where she is currently working toward the M.S. degree in the media IC and system laboratory, Graduate Institute of Electronics Engineering.

Her research interests include multimedia analysis, machine learning, and associated VLSI architectures.

Shao-Yi Chien (S'99–M'04) received the B.S. and Ph.D. degrees from the Department of Electrical Engineering, National Taiwan University (NTU), Taipei, in 1999 and 2003, respectively.

During 2003 to 2004, he was a research staff in Quanta Research Institute, Tao Yuan Shien, Taiwan. In 2004, he joined the Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, as an Assistant Professor. Since 2008, he has been an Associate Professor. His research interests include video segmentation algorithm, intelligent video coding technology, image processing, computer graphics, and associated VLSI architectures.

Daisuke Deguchi received B.S. and M.S. degrees in engineering, and the Ph.D. degree in information science from Nagoya University, Nagoya, Japan, in 2001, 2003, and 2006, respectively.

He is currently an Assistant Professor at the Graduate School of Information Science, Nagoya University. He is working on the object detection, segmentation, recognition from videos, and their applications to ITS technologies, such as the detection and the recognition of traffic signs.

Ichiro Ide received the B.Eng., M.Eng., and Ph.D. degrees from the Department of Engineering, The University of Tokyo, Tokyo, Japan, in 1994, 1996, and 2000, respectively.

During 2000 to 2004, he was an Assistant Professor at the National Institute of Informatics, Japan. Since 2000, he has been an Associate Professor at the Graduate School of Information Science, Nagoya University, Japan. During 2010 and 2011, he was also a Senior Visiting Researcher at the University of Amsterdam, The Netherlands. His research interests include multimedia contents analysis and authoring, and also multimedia support for daily life, especially for cooking and eating activities.

Hiroshi Murase (M'87–SM'00–F'06) received the B.S., M.S., and Ph.D. degrees in electrical engineering from Nagoya University, Japan, in 1978, 1980, and 1987, respectively.

In 1980 he joined the Nippon Telegraph and Telephone Corporation (NTT). From 1992 to 1993 he was a visiting research scientist at Columbia University, New York. From 2001 to 2003, he was Executive Manager of NTT Communication Science Laboratories. He is currently a Professor at the Graduate School of Information Science, Nagoya University. His research interests include computer vision, pattern recognition, multimedia information recognition, and ITS applications.

Dr. Murase was awarded the IEEE CVPR Best Paper Award in 1994, the IEEE ICRA Best Video Award in 1996, the IEEE Transaction on Multimedia Paper Award in 2004, etc. He is a fellow of the IEICE Japan.