A High-Performance VLSI Architecture for the Histogram Peak-Climbing Data Clustering Algorithm

Orlando J. Hernandez, *Member, IEEE*

***Abstract*—Image feature separation is a crucial step for image segmentation in computer vision systems. One efficient and powerful approach is the unsupervised clustering of the resulting data set; however, it is a very computationally intensive task. This paper presents a high-performance architecture for unsupervised data clustering. This architecture is suitable for VLSI implementations. It exploits paradigms of massive connectivity like those inspired by neural networks, and parallelism and functionality integration that can be afforded by emerging nanometer semiconductor technologies. By utilizing a "global-quasi-systolic, local-hyper-connected" architectural approach, the hardware can process real-time DVD-quality video at the highest rate allowed by the MPEG-2 standard. The architecture is a realization of the histogram peak-climbing clustering algorithm, and it is the first special-purpose architecture that has been proposed for this important problem. The architecture has also been prototyped using a Xilinx field programmable gate array (FPGA) development environment. Although this paper discusses a computer vision application, the architecture presented can be utilized in the acceleration of the clustering process of any type of high-dimensionality data.**

***Index Terms*—Data clustering, high-performance architectures, peak climbing algorithm, specialized architectures, VLSI design.**

I. Introduction

As NEW algorithms are developed using a paradigm of off-line non-real-time implementation, often there is a need to adapt and advance the state of the art of hardware architectures to implement such algorithms in a real-time manner if they are to truly serve a useful purpose in industry and defense, and beyond an academic setting. Such is the case with many underlying algorithms used in computer vision. Image feature separation is an important step for image segmentation in computer vision systems. One efficient and powerful approach to solving this problem is the unsupervised clustering of the resulting data set; however, it is a very computationally intensive task. This paper presents a high-performance architecture for the task of unsupervised data clustering, and preliminary results of this research were presented in [1].

The paper presents the mapping of the unsupervised histogram peak-climbing clustering algorithm to a novel high-speed architecture suitable for VLSI implementation and real-time performance. Specifically, this architecture exploits paradigms of massive connectivity like those inspired by neural networks, and parallelism and functionality integration that can be afforded by emerging nanometer semiconductor technologies. Special attention is paid to the clustering of high-dimensional sparse data sets like those found in the clustering of information-rich features used for color texture-based image segmentation [2]. By utilizing a "global-quasi-systolic, local-hyper-connected" architectural approach, the hardware can process real-time DVD-quality video at the highest rate allowed by the MPEG-2 standard. The architecture has also been prototyped using a Xilinx field-programmable gate array (FPGA) development environment. Although this paper discusses a computer vision application, the architecture presented can be utilized in the acceleration of the clustering process of any type of high-volume data, such as in the analysis of gene expression [3]–[5], genomes [6], genetic sequences [7], and data mining [8], [9].

A. Image Segmentation

The proposed architecture is of special interest for the image segmentation application to computer-vision tasks. The target segmentation algorithm, which is described in detail in [2], relies on scanning images or video frames with a sliding window and extracting features from each window. The windowing operation consists of sliding a window from left to right and top to bottom across the image. The texture, or the pixels bounded by each window, is characterized using mathematically modeled features. Once all features are extracted from the sliding windows, they are clustered in the feature space. The mapping of the identified clusters back into the image domain results in the desired segmentation.

B. Clustering for Image Segmentation

Clustering algorithms can be implemented in conventional computer platforms. While these can have a high degree of flexibility, they do carry the burden of not being tuned specifically for the task of clustering, and, therefore, suffer from a high amount of overhead and are inherently inefficient. The clustering algorithm mapped in this work has been benchmarked as requiring 193 ms running on a 2.27-GHz processor with virtually infinite memory (RAM—not disk) when clustering 961 vectors of 22 dimensions each. This data resulted from an image resolution of 128×128 pixels. If the difference in resolution is accounted for (576×702 pixels for DVD-quality video), this performance is 3879-fold below the level necessary to sustain MPEG-2 video rates of 30 frames/s, which is attained by the proposed architecture.

Manuscript received March 15, 2005; revised July 11, 2005.

The author is with the Image Processing and Understanding Laboratory, and the Computer Architecture and VLSI Laboratory, Department of Electrical and Computer Engineering, The College of New Jersey, Ewing, NJ 08628 USA (e-mail: hernande@tcnj.edu).

Digital Object Identifier 10.1109/TVLSI.2005.863761

C. Organization of the Paper

The organization of the rest of this paper is as follows. Section II discusses related works, Section III describes the clustering algorithm in detail, Section IV presents the overall architecture, and Section V describes the details. Section VI discusses the FPGA prototyping, validation, and test, along with hardware cost, and finally, Section VII offers conclusions and directions for further research.

II. Recent Related Work

Significant advances in the quality of color image segmentation results have recently been reported in the literature [2], [10]. This methodology uses high-dimensional multispectral random field texture models [11], [12] and color content as features of a subimage defined by a sliding window. These features are in turn clustered using an unsupervised peak-climbing algorithm in the highly multidimensional feature space. Once the features are clustered, these clusters are mapped back to the spatial domain of the image, which results in the image segmentation.

Although great effort has been devoted to devising hardware architectures to accelerate the execution of clustering algorithms, these efforts have not fully addressed the high-performance demands of real-time high-quality color video processing. Our proposed architecture enables the presence of clustering functionality in embedded computer vision processors and systems that have to process high-quality real-time video data. A recent clustering architecture that is fit for VLSI implementation is proposed in [13]. It uses a new adaptive κ-means clustering algorithm for real-time video imaging. In the proposed solution, a weighted contribution of both pixel intensity and distance between the pixels is used for cluster identification. The algorithm was implemented with 15K gates, and simulation results prove that a QCIF image could be handled at 15 frames/s. While the low gate count of this implementation is impressive, when compared to our proposal, it does not address color imagery, achieves half the frame rate, and the resolution is much smaller. QCIF resolution is either 176×120 or 176×144 pixels, depending on whether the context is NTSC or PAL video, and it is at most 6.27% of the DVD-quality resolution that we address (576×702 pixels). The specific problem of conceiving architectures for the efficient unsupervised peak-climbing clustering algorithm has not been addressed either. The scheme proposed in [14] uses artificial neural networks (ANNs) for cluster separation, but this implementation, being ANN based, inherently needs training (i.e., is supervised), and it only addresses a three-cluster problem. Our proposal can handle a range of clusters, from 1 to 24 882. Other architectures reported have been of an analog nature. In [15], the authors present a mixed-mode VLSI chip performing unsupervised clustering and classification, implementing models of fuzzy adaptive resonance theory (ART) and learning vector quantization (LVQ), and extending to variants such as Kohonen self-organizing maps (SOMs). The parallel processor classifies analog vectorial data into a digital code in a single clock, and implements on-line learning of the analog templates, stored locally and dynamically using the same adaptive circuits for on-chip quantization and refresh. When compared with our proposal, this prototype architecture can only handle eight inputs, and can discriminate among only 16 categories. The architecture proposed in this paper can handle 24 882 inputs and can output results of up to the same number of clusters. While analog processing tends to necessitate fewer components for a given complex operation, the technology and the approach suffer from several drawbacks. Namely, analog VLSI technology is more expensive to manufacture and test, is less accurate than a digital implementation, and suffers from serious sensitivity to noise, and its performance is very susceptible to changes in supply voltage and environmental temperature conditions. Compensating for all these susceptibilities and obtaining a robust design may require added complexity, which may hinder the manufacturability of the circuitry. Other related image recognition systems [16], while being able to handle high-dimensional data, still only address grayscale imagery. Other works found in the literature describe the implementation of the squared-error clustering algorithm [17]–[20], while still other efforts have been mostly focused on aiding the performance of ANNs [21]–[27], or apply to very narrow specific problem domains [28]. This is the first special-purpose architecture that has been proposed for the important problem of clustering massive amounts of feature data generated during the real-time segmentation of color imagery at high-quality video resolution and frame rates—specifically decoded MPEG-2 video at a resolution of 576×702 pixels and at 30 frames/s.

A. Motivation and Significance of the Proposed Architecture

Two major considerations drove the choice of the unsupervised peak-climbing clustering algorithm by means of the proposed architecture. First, the performance of an image segmentation procedure that employs this clustering algorithm has been reported in [2]. This segmentation scheme using the subject clustering algorithm has performed quite well on natural color scenery [29] when compared with the image segmentation results achieved using the well-known JSEG method described in [30].

The other consideration for the implementation of the clustering algorithm as a dedicated architecture is its simplicity and highly nonparametric nature where very few inputs are required into the system. These characteristics lend the algorithm to implementation in an FPGA environment, so it can form part of a flexible and reconfigurable real-time computing platform for video frame segmentation. This system is depicted in Fig. 1, and it can also serve as a rapid prototyping image segmentation platform. One to several windows used for the image feature extraction can be mapped on demand to a parallel array of digital signal processors (DSPs) depending on the complexity of the feature extraction algorithm and the computational bandwidth of the DSPs. This approach is possible because the feature extraction from the image windows is an uncorrelated process in a per-window basis that can be parallelized. However, the separation of the features into image regions requires the availability of all features, and it represents a computational bottleneck during image segmentation. A high-performance FPGA implementation is a solution to this bottleneck issue. In this system, the types of features and the clustering algorithm to be used can be changed on demand based on some measure of the quality

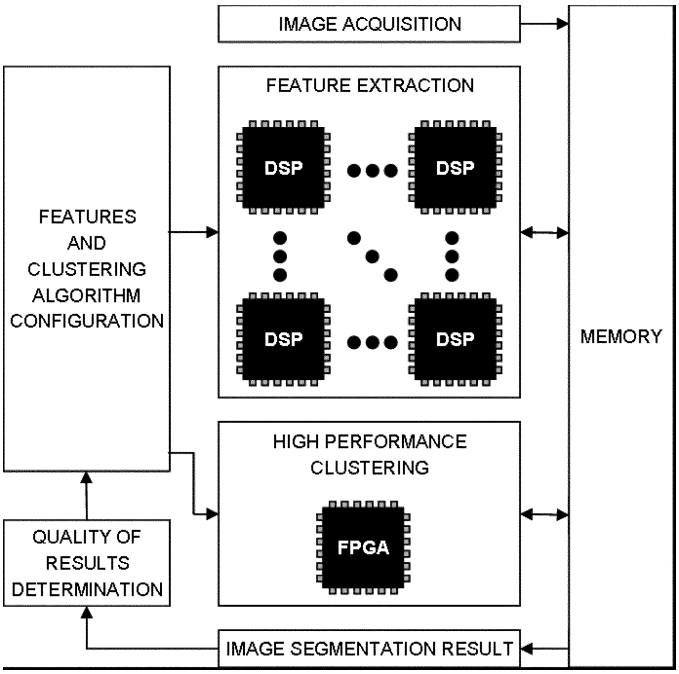

Fig. 1. Reconfigurable real-time computing platform for video frame segmentation.

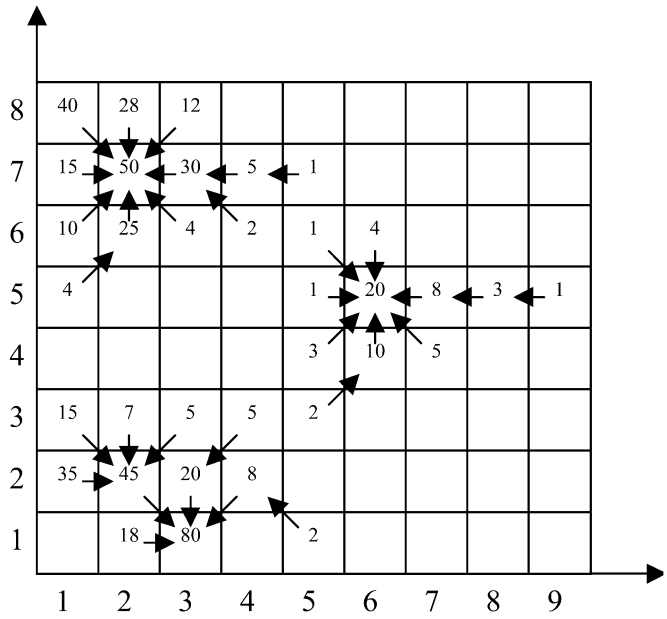

Fig. 2. Illustration of the peak climbing approach for a two-dimensional feature space example.

of results of the image segmentation task. During image segmentation, it is highly desirable to be able to choose the best fitting features and/or clustering method based on problem domain, type of imagery, or lighting conditions. The issue of cost effectiveness for FPGA implementation is somewhat secondary for reconfigurable computing platforms. The main advantage of FPGAs is flexibility, while any hardwired design, either digital or analog, will be more cost effective from a silicon-area perspective. Not withstanding, a direct normalized comparison of the proposed architecture with other works is presented in Section VI.

III. Clustering Algorithm

This section describes the clustering algorithm implemented in this work. Given M features f of dimensionality N to be clustered, the first step is to generate a histogram of N dimensions [2], [31]. This histogram is generated by quantizing each dimension according to the following equations:

$$\mathrm{CS}(k) = \frac{f_{\max}(k) - f_{\min}(k)}{Q}, \quad k = 1, 2, \ldots, N \tag{1}$$

$$d_k = \mathrm{INT}\left\{ \frac{f(k) - f_{\min}(k)}{\mathrm{CS}(k)} + 1 \right\}, \quad k = 1, 2, \ldots, N \tag{2}$$

for each of the $Mf(k)$ feature members, where

- N dimensions of the features;
- $\mathrm{CS}(k)$ length of the histogram cell in the kth dimension;
- $f_{\max}(k)$ maximum value of the kth dimension of the M features;
- $f_{\min}(k)$ minimum value of the kth dimension of the M features;
- Q total number of quantization levels for each dimension of the N-dimensional histogram;
- d_k index for a histogram cell in the kth dimension associated with a given feature **f**.

Since the dynamic range of the vectors in each dimension can be quite different, the cell size for each dimension could be different. Hence, the cells will be hyper-boxes. This provides efficient dynamic range management of the data, which tends to enhance the quality and accuracy of the results. Next, the number of feature vectors falling in each hyper-box is counted and this count is associated with the respective hyper-box, creating the required histogram.

After the histogram is generated in the feature space, a peak-climbing clustering approach is utilized to group the features into distinct clusters. This is done by locating the peaks of the histogram. In Fig. 2, this peak-climbing approach is illustrated for a two-dimensional space example.

The number in each cell (hyper-box) represents a hypothetical count for the feature vectors captured by that cell. By examining the counts of the eight neighbors of a particular cell, a link is established between that cell and the closest cell having the largest count in the neighborhood. At the end of the link assignment, each cell is linked to one parent cell, but can be a parent of more than one cell. A peak is defined as being a cell with the largest density in the neighborhood, i.e., a cell with no parent. A peak and all the cells that are linked to it are taken as a distinct cluster representing a mode in the histogram. Once the clusters are found, these can be mapped back to the original data domain from which the features were extracted. Features grouped in the same cluster are tagged as belonging to the same category.

IV. High-Performance Data Clustering Architecture

For the purpose of developing the architecture, the input data structure is defined as an address to a data tensor in memory with dimensions $J_V \times J_H \times N$ containing video frame features, and

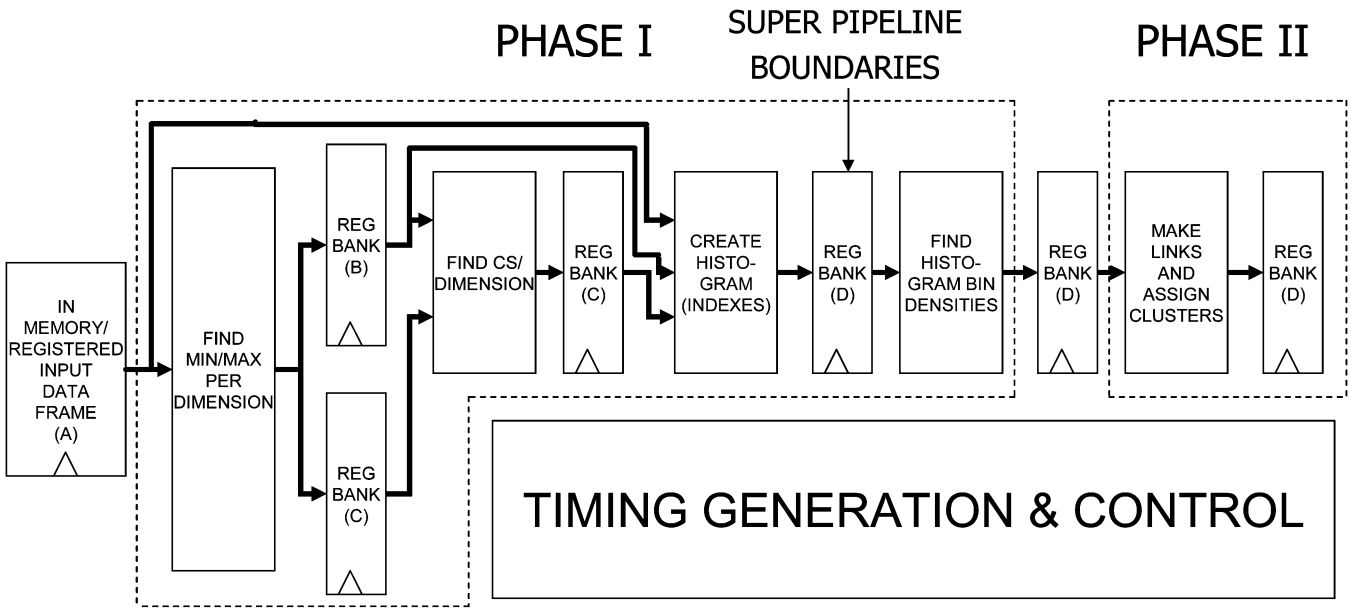

Fig. 3. Peak climbing clustering algorithm overall architecture.

the number of quantization levels to be used, which applies to all dimensions of the feature space. J_V denotes the number of spatial windows used to extract the features in the vertical direction, and J_H is the number of windows in the horizontal direction. For convenience, let us denote the total number of feature vectors as $J = J_V \times J_H$. The data set is formatted as fixed point numbers normalized in the interval $[-1, +1)$ over the entire $J \times N$ data set. These inputs are quantized to 32-b 2's complement numbers, so the input ranges from $0 \times 80000000/2^{31} = -1$ to $0 \times \mathrm{EFFFFFFF}/2^{31} = 0.9999999995$ in steps of $0 \times 00000001/2^{31} = 2^{(-31)} = 0.000000005$. This means that the numerical range of the data to be clustered (the feature vectors) is normalized from -1 inclusive to $+1 - 2^{(-31)}$ where $2^{(-31)}$ is the numerical quantization step of the fixed point format of 2's complement 32-bit numbers with one integer bit and 31 fractional bits. The resulting output of the architecture can be read as a $J_V \times J_H$ matrix containing the clusters in the spatial domain of the video frame and the number of resulting clusters.

Fig. 3 shows the different steps of this implementation of the clustering algorithm and the overall architectural organization. The chosen architecture follows a globally systolic partition, but it is folded along reusable register bank boundaries denoted by letters A–D. The major processing blocks are the computation of the minimum and maximum values of each dimension of the feature vectors, finding the cell size $\mathrm{CS}(k)$ for each k dimension, creating the histogram or assigning bin indexes to the data vectors, finding the histogram bin densities, and linking the bins and assigning the clusters. The mapping of the clusters back to the spatial domain of the video frame and the determination of how many clusters are present occurs automatically by virtue of reading the final output data by subsequent processing.

A. Algorithm Analysis

To understand the computational complexity, the load requirements for each part of the algorithm, and to aid with the balancing of the overall pipeline architecture, the algorithm was prototyped in C/C++, and 128×128 pixel images were used, along with extracted features using the multispectral simultaneous autoregressive (MSAR) random field model [2]. This model yields features of 22 dimensions ($N = 22$). The function used was

$$\texttt{int}\ ^{**}\texttt{cluster\_features}\big(\texttt{int}\ ^{***}f, \texttt{int}\ J_V, \texttt{int} J_H, \texttt{int}\ Q, \texttt{int}\ ^{*}\texttt{clusters}\big)$$

where

$\texttt{int}\ ^{***}f$	pointer to an input data tensor $J_V \times J_H \times 22$;
$\texttt{int}\ J_V$	number of feature windows in the vertical direction of the image (rows);
$\texttt{int}\ J_H$	number of feature windows in the horizontal direction of the image (columns);
$\texttt{int}\ Q$	quantization levels for each feature dimension in the feature space;
$\texttt{int}\ ^{*}\texttt{clusters}$	pointer to return the resulting number of clusters.

The dimensionality of the feature space ($N = 22$) is implied. The function returns a pointer to a matrix ($\texttt{int}\ ^{**}$) containing the clusters in the image space, and the hardware architecture was conceived to emulate the functionality of this C/C++ function. For the hardware implementation, the dataset to be clustered and Q are inputs to the system, and the J_V and J_H parameters are preset depending on the pixel dimensions of the image frames to be segmented. The clustered data and the number of clusters found are outputs of the system. The number of clusters found is not fixed; it varies depending on the data being clustered.

Other parameters that were set for the implementation of the algorithm are the feature extraction square window size $L \times L$ pixels, and the step of the overlapping sliding window in D pixels. Details of why a sliding window is used to extract the features can be found in [2]. L and D were set to 8 and 4 pixels, respectively, and the range of inputs for Q was chosen to be $[3, 8]$. These parameters were chosen based on the performance

TABLE I
RESULTS OF ALGORITHM BENCHMARKING AND ANALYSIS

		30 frames/second BUDGET			30 frames/second RETIMED BUDGET			
STEP	ABS TIME (mS)	TIME	128x128 (μS)	MPEG-2 (μS)	TIME	128x128 (μS)	MPEG-2 (μS)	PHASE
MIN/MAX	17	9%	3000	3000	9%	3000	3000	I
CS	17	9%	3000	3000	9%	3000	3000	
INDEXES	17	9%	3000	3000	9%	3000	3000	
DENSITY	31	16%	5332	5332	16%	5332	5332	
LINK	94	48%	15999	15999	48%	15999	15999	II
CLUSTER	17	9%	3000	3000	9%	3000	3000	
TOTAL	193	100%	33331	33331	100%	33331	33331	

of an overall higher level algorithm for a real image segmentation application [2], [10]. Thus, the architecture is scalable on the rest of the parameters, namely J_V and J_H ($J = J_V \times J_H$), which are given by the resolution of the image or video frame and N. The timing requirements are given by the frame rate of 30 frames/s, which is independent of the resolution of the image.

For any image resolution $n \times m$

$$J_V = \left\lfloor \frac{m - L}{D} \right\rfloor + 1 \tag{3}$$

and

$$J_H = \left\lfloor \frac{n - L}{D} \right\rfloor + 1 \tag{4}$$

so for the 128×128 images used for benchmarking the algorithm, $J_V = J_H = 31$ and $J = 961$, and for a DVD-quality resolution of 702×576 pixels, $J_V = 143$ and $J_H = 174$, and $J = 24\,882$. The results of the algorithm benchmarking and analysis exercise are shown in Table I, and the specific steps will be clarified in Section IV-B. The benchmark numbers are used as a guide to what parts of the algorithm consume the most time, and, therefore, which parts should be parallelized more with respect to others (i.e., parallelization along the feature space N, or both N and the number of vectors to be clustered J). Although the benchmark numbers include operating system overhead that is not present in the dedicated proposed architecture, this data proved to be useful in analyzing how the overall architecture could be optimally balanced from a timing perspective to achieve the desired results. The relative amount of time spent in a particular step or phase of the clustering operation is useful in understanding the relative complexity of the steps with respect to the others, and the inference from this data was that the hardware could be parallelized along the number of feature dimensions N, and processed sequentially along the number of feature vectors J for some steps. For other steps, the hardware had to be parallelized along both N and J algorithmic dimensions. This results in an efficient balance between performance and hardware complexity. In other words, it produces a minimum-size processor that can cluster features for the segmentation of DVD-resolution video (i.e., 702×576 pixels) in real time at 30 frames/s.

B. Hardware Algorithm

The mapped hardware algorithm performed by the architecture is shown as follows.

```
Hardware Algorithm
input: Feature vectors, pre-normalized and
quantized.
output: Map of clustered data vectors, and number
of clusters.
begin
// MIN/MAX STEP
do in parallel N times
parbegin
 f_min(k) ← f_0(k)
 for (i = 0; i < J; i++) begin
  if (f_i(k) < f_min(k)) f_min(k) ← f_i(k)
 end
 f_max(k) ← f_0(k)
 for (i = 0; i < J; i++) begin
  if (f_i(k) > f_max(k)) f_max(k) ← f_i(k)
 end
parend
// CS STEP
for (i = 0; i < N; i++) begin
 CS (k) ← (f_max(k) − f_min(k))/Q
end
// INDEXES STEP
do in parallel N times
parbegin
 for (i = 0; i < J; i++) begin
  do in parallel
  parbegin
   d_k ← (int)((f_i(k) − f_min(k))/CS(k))
   histogram[i].linkto_coordinates[k] ←
    (int) ((f_i(k) − f_min(k))/CS(k))
  parend
 end
parend
// FIND HISTOGRAM BIN DENSITIES STEP
for (i = 0; i < J; i++) begin
 allocated ← FALSE
 equal ← TRUE
 do in parallel J times
 parbegin
  do in parallel N times
  parbegin
   if (d_k ≠ histogram[i].coordinates[k])
    equal ← FALSE
  parend
  if (equal and not allocated) then
```

```
   do in parallel
   parbegin
    allocated ← TRUE
    histogram[i].cell_count ←
     histogram density
   parend
  endif
 parend
end
// LINKS AND CLUSTER STEPS
do in parallel J times with index variable i
parbegin
 histogram[i].linkto_cell_count ←
  histogram[i].cell_count
for (ii = 0; ii < J − 1; ii++) begin
   if (ii ≠ i) then
    neighbor ← TRUE
    do in parallel N times
    parbegin
     if (abs(histogram[i].coordinates[k]
      −histogram[ii].coordinates[k]) > 1)
      neighbor ← FALSE
   parend
   if (neighbor) then
   if (histogram[ii].cell_count >
    histogram[i].linkto_cell_count) then
    do in parallel N times
    parbegin
     histogram[i].
     linkto_coordinates[k] ←
      histogram[ii].coordinates[k]
   parend
    histogram[i].linkto_cell_count ←
     histogram[ii].cell_count
   endif
  endif
 endif
end
for (ii = J − 1; ii >= 0; ii − −) begin
 if (ii ≠ i) then
  neighbor ← TRUE
  do in parallel N times
  parbegin
   if ((abs(histogram[i].coordinates[k]
    − histogram[ii].coordinates[k]) > 1)
    neighbor ← FALSE
  parend
  if (neighbor) then
   if (histogram[ii].cell_count >
histogram[i].linkto_cell_count) then
     do in parallel N times
     parbegin
      histogram[i].linkto_coordinates[k] ←
       histogram[ii].coordinates[k]
     parend
     histogram[i].linkto_cell_count ←
      histogram[ii].cell_count
    endif
   endif
  endif
 end
parend
end
```

C. Performance Gain Analysis

By observing the number of major computational loops for the implementation in a generic computer platform, the level of complexity of the clustering algorithm is $O(J^2)$. Formally,

$$CO_{\mathrm{GEN}}(N, J_V, J_H) = 3 \times N \times J_V \times J_H + N + 5 \times N \times (J_V \times J_H)^2 \quad (5)$$

and

$$CO_{\mathrm{GEN}}(N, J) = 3 \times N \times J + N + 5 \times N \times J^2 \quad \forall J = J_V \times J_H. \quad (6)$$

On the other hand, the complexity of the proposed architecture is $O(J)$. Specifically

$$CO_{\mathrm{NEW}}(N, J_V, J_H) = 6 \times J_V \times J_H + N - 1 \quad (7)$$

and

$$CO_{\mathrm{NEW}}(N, J) = 6 \times J + N - 1 \qquad \forall J = J_V \times J_H. \quad (8)$$

Note that the complexity of the new architecture does not depend on the number of features N because of the complete parallelization along this algorithmic dimension N. Thus the performance gain realized by the proposed architecture is $O(J)$

$$\mathrm{SPEEDUP} = \frac{CO_{\mathrm{GEN}}(N, J)}{CO_{\mathrm{NEW}}(N, J)} = \frac{3 \times N \times J + N + 5 \times N \times J^2}{6 \times J + N - 1}. \quad (9)$$

Since $N \ll J$, $J \ll J^2$, and $6 \ll J$

$$\mathrm{SPEEDUP} \cong \frac{J^2}{J} = J. \quad (10)$$

Using the benchmark timing numbers in the general-purpose implementation, the minimum speedup that the proposed architecture is required to achieve over the general purpose implementation to support DVD-resolution video at 30 frames/s can be calculated as follows:

$$\mathrm{SPEEDUP}_{\mathrm{REQ}} = \frac{T_{\mathrm{GEN}(128\times128)} \times \frac{CO_{\mathrm{GEN}}(N, J_{\mathrm{DVD}})}{CO_{\mathrm{GEN}}(N, J_{128\times128})}}{T_{\mathrm{MIN}}} \quad (11)$$

$$\mathrm{SPEEDUP}_{\mathrm{REQ}} = \frac{T_{\mathrm{GEN}(128\times128)} \times \left(\frac{3\times N\times J_{\mathrm{DVD}} + N + 5\times N\times J_{\mathrm{DVD}}^2}{3\times N\times J_{128\times128} + N + 5\times N\times J_{128\times128}^2}\right)}{T_{\mathrm{MIN}}} \quad (12)$$

$$\mathrm{SPEEDUP}_{\mathrm{REQ}} = \frac{193mS \times \left(\frac{3\times22\times24\,882 + 22 + 5\times22\times24\,882^2}{3\times22\times961 + 22 + 5\times22\times961^2}\right)}{\frac{1}{30}S} = 3879. \quad (13)$$

This is well within the upper bound of speedup $J = 24\,822$ that can be achieved due to architectural and parallel VLSI implementation speed gains with the newly proposed architecture.

V. Architectural Details

This section presents the architectural details of the processor. In all figures, the processing element (PE) being discussed is bounded by dashed lines. To perform the operation of finding the minimum and maximum values for each dimension of the feature vectors in the data set, N Min-Max PEs are instantiated in parallel, one for each dimension. The operations to find the minimum and maximum values are run sequentially, thus, making use of a single MIN/MAX cell in the PE. Each instantiated PE cycles $2 \times J$ times through all the values in a given dimension of the feature vectors.

Since the number of times that the process to compute the cell size $\mathrm{CS}(k)$ for each dimension needs to be executed depends only on N, and not on J, which dominates the complexity of the algorithm; only one PE is instantiated for this task, and the process is executed N times. One property of this architecture is that it is tuned to high-dimensional features. Analysis of the behavior of the clustering algorithm reveals that it is sufficient to quantize the multidimensional histogram into only very few levels to attain effective feature separation. Because of the high number of dimensions, the number of resulting bins in the histogram N^Q can be much larger than the number of feature vectors to be clustered. This has the effect of forcing the data being clustered to be organized sparsely in the histogram, which aids in feature separation and effective clustering. Specifically, for the Random Field model features utilized during the benchmarking exercise, the number of quantization levels necessary $Q = 3, \ldots, 8$, and the architecture makes use of this constraint. This allows the division operation of (1) to be implemented by a multiplication by the inverse of Q stored in a small look-up table (LUT) of only six entries. For the specific case of DVD-resolution video (i.e., 702×576 pixels) with $J_V = 143$ and $J_H = 174$, and $N = 22$ for the Random Field model, the number of feature vectors is $J = 24\,882$, and the number of bins in the histogram range from 22^3 to 22^8; this is from 10 648 to more than 54 billion. The number of data vectors can indeed be small relative to the number of bins in the histogram. The data format for Q_inv and the resulting CS are also formatted like the feature data (i.e., MIN and MAX for this PE). The format is 32-bit 2's complement fixed-point numbers with a 1-bit integer part and a 31-bit fractional part.

To compute the histogram indexes for each data vector, N PEs are instantiated in parallel; one for each dimension, and each instantiated PE cycles J times through all the values in a given dimension of the feature vectors. Since the possible number of quantization levels has been constrained to six, the division in the Index PE can be implemented by a simple parallel restoring division algorithm that has been limited to computing only the three most significant bits of the quotient. Since Q only takes values from three to eight (six possible values), the bin index for any given dimension can be coded with 3 bits, and, therefore, the division during the computation of the bin indexes is limited to only the three most significant bits. The results of this operation will populate the histogram indexes in the storage data structure, and the "`linkto_coordinates`" data structure will also be populated with the initial indexes of the data. Since no clustering has occurred yet, it is only established that each data vector only has a link to itself.

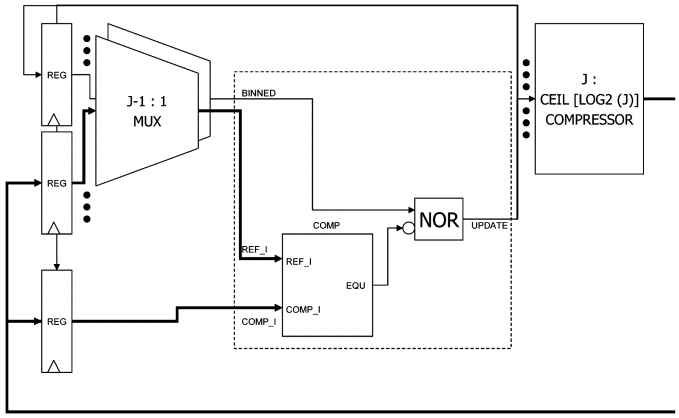

Fig. 4. Processing element used to identify histogram bin density.

Fig. 4 shows the details of the PE to identify a data vector with a given histogram bin density. J instantiations of this PE are made, which corresponds to one instantiation per each possible bin. Then this process runs J times to compare the coordinates of every vector with those of every other vector, and determine those that are in the same histogram bin, and, thus, the density of the nonempty histogram bins. The purpose of the compressor is to count in parallel the number of ones from the comparators, which corresponds to the density of a given bin in the histogram. There is also a flag that signifies whether a specific data vector has been binned, and it is initially cleared for all vectors, meaning that the vectors have not been binned. If a vector has not been binned and its histogram coordinates are equal to any of the vectors with which it is being compared, then the density of the histogram bin "`cell_count`" is stored, and the bin coordinates are used as a unique bin identifier label. Table II additionally shows an example of the specific structure of half adders and full adders for the compressor tree in the case of DVD-quality resolution where the maximum output value of the compressor is $J = 24\,882$.

The micro-architecture to establish the links between the histogram bins and cluster the vectored data is shown in Fig. 5 and the PE is shown in Fig. 6. This process is executed $(J-1)$ times from the low to high direction of the linear vector address space, and then for all possible bins J from the high to low direction of the linear vector address space. This has the effect of linking and clustering the data by using the bin to which a data vector is linked as the label for the clusters. After these two passes, the "`linkto_coordinates`" fields for all vectors in the same cluster will contain the value of the coordinates of the bin that represents a mode or a peak in the multidimensional histogram, and, therefore, can be use as a cluster identifier. Using the vectors as a linear list removes the need of considering the histogram as a multidimensional data structure, and takes advantage of sparse histograms, which are typical of image segmentation applications that use high-dimensional features. With this scheme, only the vectors that are being clustered are processed,

TABLE II
COMPRESSOR STRUCTURE

MODULE DESCRIPTION	MODULE NAME	OPERATION ON PREVIOUS LAYER/STAGE	NUMBER OF MODULES	IN	OUT	MAXIMUM OUTPUT VALUE
INPUTS	INPUTS	NONE	24882	24882b	1b	1
FA	fa	/3	8294	3b	2b	3
HA	ha_2b	/2	4147	2b	3b	6
HA	ha_3b	/2, round down	2073	3b	4b	12
HA	ha_4b	/2, round up	1037	4b	5b	24
HA	ha_5b	/2, round down	518	5b	6b	48
HA	ha_6b	/2	259	6b	7b	96
HA	ha_7b	/2, round up	130	7b	8b	192
HA	ha_8b	/2	65	8b	9b	384
HA	ha_9b	/2, round down	32	9b	10b	768
HA	ha_10b	/2	16	10b	11b	1536
HA	ha_11b	/2	8	11b	12b	3072
HA	ha_12b	/2	4	12b	13b	6144
HA	ha_13b	/2	2	13b	14b	12288
HA	ha_14b	/2	1	14b	15b	24576
HA	ha_15b	/2, round up	1	15b	15b	24882[a]

[a]Capped at the maximum value of J = 24,882.

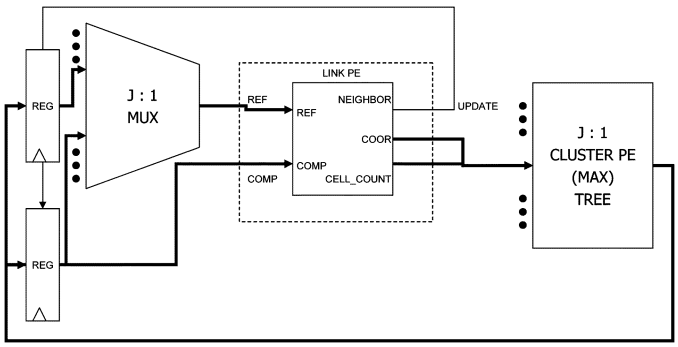

Fig. 5. Processing element used to link and cluster data.

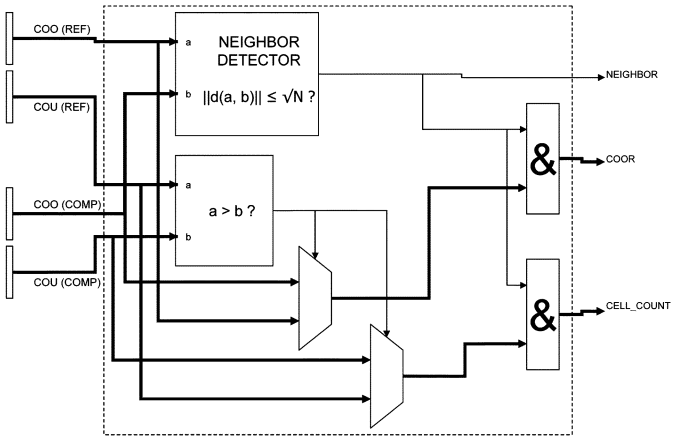

Fig. 6. Link and cluster processing element.

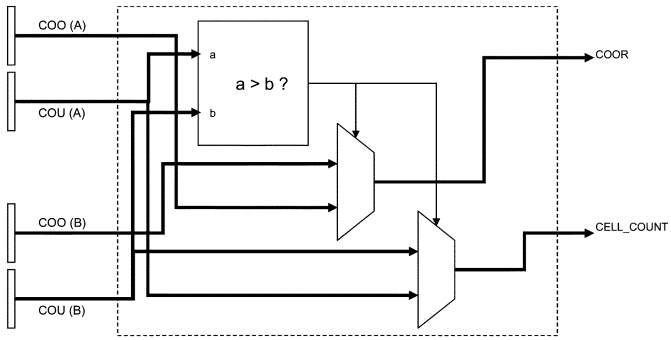

Fig. 7. MAX cluster processing element.

and the empty bins in the histogram are never considered. The PE uses a novel computational cell to calculate the norm between two N-dimensional vectors, and determine whether a data vector is a neighbor of another. Regardless of the dimension of the data, it is only sufficient to determine whether the distance between the vectors is greater than one in at least one dimension to determine if the vectors are neighbors. Therefore, this cell implicitly computes a norm bound between two high-dimension vectors $||d(a,b)|| \leq \sqrt{N}$, and determines the neighborhood property. When comparing a data vector with all others, the density of the bin where each data vector lays is routed through a tree that finds the coordinates of the data bin with the highest density. This will be a peak or mode in the multidimensional histogram, and it becomes the cluster label for all neighboring data vectors. All neighboring data vectors inherit the coordinates and the density associated with the neighbor vector that is in the bin with the highest density as given by "`cell_count`." This information is populated into the "`linkto_coordinates`" and the "`cell_count`" storage data structures. The circuit for the maximum finding structure is shown in Fig. 7. The mapping back to the spatial coordinate system of the image takes place automatically by virtue of reading the final result from the local storage of the overall architecture, as is the determination of the number of clusters present in the segmented image or video frame. Finally, a global clocking, control, and shared storage network tie all the modules together to form the complete architecture, as shown in Fig. 3.

VI. FPGA PROTOTYPING, VALIDATION, AND TEST

The proposed architecture was implemented in Verilog for both a DVD-quality resolution of 576 $\times$702 pixels and a resolution of 128 $\times$128 pixels. Both configurations were synthesized using the Artisan library for IBM's 0.13-μm silicon process obtained through MOSIS [32]. These designs were also simulated using the Mentor Graphics Corporation hardware simulation tools to verify that the hardware performed as expected and at a video rate of 30 frames/s. Additionally, the architecture was implemented in an FPGA environment

TABLE III
PERFORMANCE DETAILS

ARCHITECTURE	PROPOSED			Maliatski & Yadid-Pecht [13]
COLOR	Y	Y	Y	N
RESOLUTION	128x128	128x128	576x702 (DVD)	176x144 (QCIF)
IMPLEMENTATION	FPGA	STANDARD CELL	STANDARD CELL	STANDARD CELL
$T_{MIN/MAX}$ (nS)	5.264	7.997	7.997	
$F_{MIN/MAX}$ (MHz)	189	125	125	
T_{CS} (nS)	22.413	22.507	22.507	
F_{CS} (MHz)	44	44	44	
$T_{INDEXES}$ (nS)	18.041	24.086	24.086	
$F_{INDEXES}$ (MHz)	55	41	41	
$T_{DENSITIES}$ (nS)	26.873	26.142	39.827	
$F_{DENSITIES}$ (MHz)	37	38	25	
$T_{LINK/CLUSTER}$ (nS)	57.970	94.108	133.628	
$F_{LINK/CLUSTER}$ (MHz)	17	10	7	
$T_{F\text{-}MIN}$ (mS)	0.165	0.245	8.638	66.667
MAX frames/S	6055	4083	115	15
T (nS)	57.970	94.108	133.628	12.500
F (MHz)	17	10	7	80
F_{MIN} (MHz)	0.174	0.174	4.479	80
$T_{F\text{-}NOM}$ (mS)	0.335	0.545	19.952	66.667
NOM frames/S	2980	1836	50	15

for a video resolution of 128 ×128 pixels and 30 frames/s. To this end, a rapid prototyping system that combines a Texas Instruments Inc. TMS320C6701 floating-point digital signal processor (DSP) with a peak performance of 1 GFLOPS and a Xilinx FPGA (Virtex-II XC2V1000) was employed. The DSP was used to extract and preprocess MSAR features from video frames, and the proposed architecture mapped to the FPGA was used to cluster the features, thus completing the video segmentation task.

A. Performance

The minimum time required to process a frame in terms of the processing time for each step is given by the expression

$$T_{F(R)-\mathrm{MIN}} = 2 \times J \times T_{\mathrm{MIN/MAX}(R)} + N \times T_{\mathrm{CS}(R)} + J \times T_{\mathrm{INDEXES}(R)} + J \times T_{\mathrm{DENSITIES}(R)} + (2J-1) \times T_{\mathrm{LINK/CLUSTER}(R)} \quad (14)$$

where R denotes the resolution of the frame. This $T_{F(R)-\mathrm{MIN}}$ will yield the maximum frame rate that the architecture can sustain for a given silicon technology implementation and for a given frame resolution R when running with multiple clock domains for the different steps. However, if all T's are equalized to (15), shown at the bottom of the page, then

$$T_{F(R)-\mathrm{NOM}} = (6 \times J + N - 1) \times T_{(R)} \quad (16)$$

where $T_{F(R)-\mathrm{MIN}}$ is the single slower frequency $F_{(R)} = (1/T_{(R)})$ at which the architecture can run while achieving a lower frame rate, but still equal or above 30 frames/s. For a given resolution, the minimum frequency to achieve 30 frames/s is

$$F_{(R)-\mathrm{MIN}} = 30 \times (6 \times J = N - 1) \quad (17)$$

which will be achieved as long as $T_{(R)} \leq T_{(R)-\mathrm{MIN}}$, where

$$T_{(R)-\mathrm{MIN}} = \frac{1}{F_{(R)-\mathrm{MIN}}}.$$

Table III shows different aspects of the performance of the proposed architecture, and compares it with the latest proposed architecture for clustering found in the literature [13]. In this table, the various maximum frequencies for the different steps are shown, the maximum frame rates achievable with multiple clock domains in the implementations attempted, the frequency and frame rates attainable if the system runs with a single clock, and the minimum frequencies that the architecture needs to run in single clock mode to achieve at least 30 frames/s. It has been shown that the architecture can always run above this minimum frequency with a frame rate greater than 30 frames/s. Our target was to achieve this frame rate for the high-quality DVD-resolution of 576 ×702 pixels, and this has been achieved. In contrast, the architecture described in [13] achieves a frame rate of 15 frames/s for the much lower QCIF resolution, and does not encompass processing of color imagery.

B. Hardware Cost

The storage hardware cost for this architecture is given in Table IV, and the totals are given by the following equations:

$$\text{TOTAL MEMORY bits} = N \times J \times 32 \quad (18)$$

$$\text{TOTAL REGISTER bits} = N \times 32 + J \times \left(2 \times 3 \times N + 1 + \lceil \mathrm{LOG}_2 \mathrm{J} \rceil\right). \quad (19)$$

$$T_{(R)} = \mathrm{MAX}\left(T_{\mathrm{MIN/MAX}(R)}, T_{\mathrm{CS}(R)}, T_{\mathrm{INDEXES}(R)} T_{\mathrm{DENSITIES}(R)}, T_{\mathrm{LINK/CLUSTER}(R)}\right) \quad (15)$$

TABLE IV
REGISTER AND MEMORY SYSTEM

MEMORY/REGISTER GROUP	DESCRIPTION
A	N blocks of J x 32b 1-port synchronous memories
B and C	N x 32b register bank
D	J x (2 x 3 x N + 1 + CEILING [LOG_2 J])b register bank

TABLE V
HARDWARE COST

	SIZE	
	128 x 128 pixels	576 x 702 pixels
MEMORY/REGISTER GROUP		
A	22 blocks of 961 x 32b 1-port synchronous memories	22 blocks of 24,882 x 32b 1-port synchronous memories
B and C	22 x 32b register bank	22 x 32b register bank
D	961 x 143b register bank	24,882 x 148b register bank
TOTAL bits OF MEMORY	676,544	17,516,928
TOTAL bits OF REGISTERS	138,831	3,683,944
EQUIVALENT LOGIC GATES		
MIN/MAX	2,211.0	2,211.0
CS	874.0	874.0
INDEX	3,718.0	3,718.0
DENSITY	77,091.0	1,997,412.5
LINK AND CLUSTER	252,094.5	7,364,930.5
CONTROLLER	8,399.7	9,967.2
TOTAL LOGIC GATES	344,388.2	9,379,113.2

As described previously, for the 576×702 pixels test case, $J = 24\,882$. Then

$$\text{TOTAL MEMORY bits} = 22 \times 24\,882 \times 32 = 17\,516\,928 \text{ bits} \quad (20)$$

and

$$\text{TOTAL REGISTER bits} = 22 \times 32 + 24\,882 \times \left(2 \times 3 \times 22 + 1 + \lceil \text{LOG}_2\ 24\,882 \rceil\right) = 3\,683\,240 \text{ bits}. \quad (21)$$

For the 128×128 pixels test case, $J = 961$. Then

$$\text{TOTAL MEMORY bits} = 22 \times 961 \times 32 = 676\,544 \text{ bits} \quad (22)$$

and

$$\text{TOTAL REGISTER bits} = 22 \times 32 + 961 \times \left(2 \times 3 \times 22 + 1 + \lceil \text{LOG}_2 961 \rceil\right) = 138\,831 \text{ bits}. \quad (23)$$

The amount of storage for the 128×128 pixels case is well within the capabilities of the XC2V1000 FPGA which contains 880 kB of RAM, and the breakdown of the storage requirements are shown in Table V for both resolutions. Table V also shows the gate count from both cases from the synthesis results. Emerging nanometer range silicon technologies are able to integrate as much as 64 million logic gates and 100 Mbits of 6-transistor SRAM. With these types of technologies, a chip of the architecture presented in this paper for the DVD-quality resolution of 576×702 pixels can be estimated to be about 100 mm^2 or 1 cm on a side, which is well within manufacturability limits. In general, the hardware cost and complexity scales with J, given that the hardware cost is linearly and directly proportional with the area of the image frame being segmented, since J is a measure of the image area.

VII. CONCLUSION

This paper proposes a high-performance special-purpose VLSI architecture for the real-time clustering of high-dimensional feature data extracted from images or video frames. The architecture has been analyzed in terms of hardware cost and prototyped in a FPGA environment, and it has been shown that processing rates suitable for DVD-quality video at MPEG-2 frame rates can be sustained, that is, 702×576 pixels at 30 frames/s. By using a top-level quasi-systolic architectural partitioning scheme and extensive connectivity at the lower levels, the performance is improved 3879-fold with respect to what can be achieved in a generic compute platform. This architecture can be used in many military, industrial, and commercial applications that require real-time intelligent machine vision processing of high-quality video. However, the approach is not limited to this type of signal processing only, but it can also be applied to other types of data for other problem domains, for which the clustering process needs to be accelerated. There is room for further performance improvements by making the architecture truly systolic. In this case, with only augmenting the amount of storage and simplifying the controller, multiple data streams could be processed simultaneously. This topic is part of our ongoing research, as is the possibility of processing data in floating-point format, and the implementation of the architecture across several FPGA chips to address larger resolutions.

ACKNOWLEDGMENT

The author would like to thank the reviewers for their helpful and insightful comments.

REFERENCES

[1] O. J. Hernandez, "High performance VLSI architecture for data clustering targeted at computer vision," in *Proc. IEEE SoutheastCon*, Apr. 2005, pp. 99–104.

[2] A. Khotanzad and O. J. Hernandez, "Color image retrieval using multispectral random field texture model and color content features," *Pattern Recognit. J.*, vol. 36, pp. 1679–1694, Aug. 2003.

[3] A. W.-C. Liew, L. K. Szeto, S.-S. Tang, H. Yan, and M. Yang, "A computational approach to gene expression data extraction and analysis," *J. VLSI Signal Process. Syst. Signal, Image, Video Technol.*, vol. 38, pp. 237–258, Nov. 2004.

[4] L. Conde, A. Mateos, J. Herrero, and J. Dopazo, "Improved class prediction in DNA microarray gene expression data by unsupervised reduction of the dimensionality followed by supervised learning with a perceptron," *J. VLSI Signal Process. Syst. Signal, Image, Video Technol.*, vol. 35, pp. 245–253, Nov. 2003.

[5] Z. Wang, Y. Wang, J. Lu, S.-Y. Kung, J. Zhang, R. Lee, J. Xuan, J. Khan, and R. Clarke, "Discriminatory mining of gene expression microarray data," *J. VLSI Signal Process. Syst. Signal, Image, Video Technol.*, vol. 35, pp. 255–272, Nov. 2003.

[6] C.-H. Huang and S. Rajasekaran, "Parallel pattern identification in biological sequences on clusters," *IEEE Trans. Nanobiosci.*, vol. 2, no. 1, pp. 29–34, Mar. 2003.

[7] S. Mukhopadhyay, C. Tang, J. Huang, and M. Palakal, "Genetic sequence classification and its application to cross-species homology detection," *J. VLSI Signal Process. Syst. Signal, Image, Video Technol.*, vol. 35, pp. 273–285, Nov. 2003.

[8] R. Tagliaferria, G. Longoc, L. Milanoc, F. Acernesec, F. f. Baronee, A. Ciaramella, R. De Rosac, C. Donalek, A. Eleuterie, G. Raiconia, S. Sessah, A. Staiano, and A. Volpicelli, "Neural networks in astronomy," *Neural Netw.*, vol. 16, pp. 297–319, 2003.

[9] I. Dhillon, J. Fan, and Y. Guan, "Efficient clustering of very large document collections," in *Data Mining for Scientific and Engineering Applications*, R. Grossman, G. Kamath, and R. Naburu, Eds. Norwell, MA: Kluwer, 2001.

[10] O. J. Hernandez, "Color image retrieval using multispectral random field texture model and color content features," Ph.D. dissertation, Southern Methodist Univ., Dallas, TX, 2002.

[11] J. W. Bennett and A. Khotanzad, "Multispectral random field models for synthesis and analysis of color images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, pp. 327–332, Mar. 1998.

[12] J. W. Bennett, "Modeling and analysis of gray tone, color, and multispectral texture images by random field models and their generalizations," Ph.D. dissertation, Southern Methodist Univ., Dallas, TX, 1997.

[13] 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.

[14] S. R. J. R. Konduri and J. F. Frenzel, "Non-linearly separable cluster classification: An application for a pulse-coded CMOS neuron," in *Proc. Artificial Neural Networks in Engineering Conf.*, vol. 13, Nov. 2003, pp. 63–67.

[15] J. Lubkin and G. Cauwenberghs, "VLSI implementation of fuzzy adaptive resonance and learning vector quantization," *Analog Integr. Circuits Signal Process.*, vol. 30, pp. 149–157, Feb. 2002.

[16] M. Yagi and T. Shibata, "An image representation algorithm compatible with neural-associative-processor-based hardware recognition systems," *IEEE Trans. Neural Netw.*, vol. 14, no. 5, pp. 1144–1161, Sep. 2003.

[17] M.-F. Lai, C.-H. Hsieh, and Y.-P. Wu, "A VLSI architecture for clustering analyzer using systolic arrays," in *Proc. 12th IASTED Int. Conf. Applied Informatics*, May 1994, pp. 260–260.

[18] M.-F. Lai, Y.-P. Wu, and C.-H. Hsieh, "Design of clustering analyzer based on systolic array architecture," in *Proc. IEEE Asia-Pacific Conf. Circuits and Systems*, Dec. 1994, pp. 67–72.

[19] M.-F. Lai, M. Nakano, Y.-P. Wu, and C.-H. Hsieh, "VLSI design of clustering analyzer using systolic arrays," *Inst. Elect. Eng. Proc.: Comput. Digit. Tech.*, vol. 142, pp. 185–192, May 1995.

[20] M.-F. Lai and C.-H. Hsieh, "A novel VLSI architecture for clustering analysis," in *Proc. IEEE Asia Pacific Conf. Circuits and Systems*, Nov. 1996, pp. 484–487.

[21] T. Serrano-Gotarredona and B. Linarea-Barranco, "A real-time clustering microchip neural engine," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 4, no. 2, pp. 195–209, Jun. 1996.

[22] J. Sitte, T. Körner, and U. Rückert, "An analog-current mode local cluster neural net," in *Proc. IEEE 6th Int. Conf. Emerging Technologies and Factory Automation*, Sep. 1997, pp. 237–242.

[23] D. D. Zhang, "System design methodology for fuzzy clustering neural networks," in *Proc. IEEE Int. Conf. Systems, Man and Cybernetics*, vol. 2, Oct. 1996, pp. 1062–1066.

[24] D. Zhang, M. Kamel, and M. I. Elmasry, "Fuzzy clustering neural network system design and implementation," in *Midwest Symp. Circuits Syst.*, vol. 2, Aug. 1994, pp. 1381–1384.

[25] D. Zhang and S. K. Pal, "A fuzzy clustering neural networks (FCNs) system design methodology," *IEEE Trans. Neural Netw.*, vol. 11, no. 5, pp. 1174–1177, Sep. 2000.

[26] P. Poiré, Y. Savaria, H. Daniel, M.-A. Cantin, and Y. Blaquière, "Hardware/software codesign of a fuzzy ART neural clusterer: The benefits of configurable computing," *Proc. SPIE*, vol. 3526, pp. 90–96, Nov. 1998.

[27] A. Granger, Y. Blaquière, Y. Savaria, M.-A. Cantin, and P. Lavoie, "A VLSI architecture for fast clustering with fuzzy ART neural networks," in *Proc. Int. Workshop on Neural Networks for Identification, Control, Robotics, and Signal/Image Processing (NICROSP)*, Aug. 1996, pp. 117–125.

[28] M.-A. Cantin, Y. Blaquière, Y. Savaria, A. Granger, and P. Lavoie, "Implementation of the fuzzy ART neural network for fast clustering of radar pulses," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 2, May 1998, pp. 458–461.

[29] *Professional Photos CD-ROM Sampler—SERIES 200000*, Corel Corp., 1994.

[30] Y. Deng and B. S. Manjunath, "Unsupervised segmentation of color-texture regions in images and video," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 8, pp. 800–810, Aug. 2001.

[31] A. Khotanzad and A. Bouarfa, "Image segmentation by a parallel, nonparametric histogram based clustering algorithm," *Pattern Recognit. J.*, vol. 23, pp. 961–963, Sep. 1990.

[32] (2005) The MOSIS Service. [Online]. Available: http://www.mosis.org/

Orlando J. Hernandez (M'93) received the Ph.D. degree in electrical engineering from Southern Methodist University, Dallas, TX, in 2002.

He is currently an Assistant Professor of electrical and computer engineering at The College of New Jersey, Ewing. From 1993 to 2003, he was with Texas Instruments Incorporated, Dallas, TX, and with Maxim, Dallas, where he held positions in design and design management. He is currently the Director of the National Science Foundation sponsored Image Processing and Understanding Laboratory, and the Computer Architecture and VLSI Laboratory at The College of New Jersey. His research interests include color image segmentation and retrieval, computer vision, image processing, and high-performance specialized VLSI architectures to perform these tasks.