A Self-Configurable Systolic Architecture for Face Recognition System Based on Principal Component Neural Network

N. Sudha, *Senior Member, IEEE,* A. R. Mohan, *Student Member, IEEE,* and Pramod K. Meher, *Senior Member, IEEE*

***Abstract*—An efficient self-configurable systolic architecture is proposed in this paper for very large scale integration implementation of a face recognition system. The proposed system applies principal component neural network (PCNN) with generalized Hebbian learning for extracting eigenfaces from the face database. It demonstrates a recognition performance of more than 85% when evaluated on the benchmark Yale and FRGC databases containing images with varying illumination and expression. Unlike the existing face recognition systems, the proposed approach not only recognizes the faces using computed eigenfaces, but also updates eigenfaces automatically whenever the face database changes. The challenge, however, lies in hardware realization of the PCNN-based face recognition system. In the presence of computation-intensive steps of varying nature, it is not straightforward to map the overall computation to a single systolic architecture. A primary contribution of this paper from the architecture point of view is an optimized mapping of fine-grained systolized signal flow graphs (SFGs) for each individual step of the algorithm on to a single self-configurable linear systolic array by appropriate merging of the computations pertaining to different nodes of different SFGs. The architecture has the flexibility of processing face images and databases of any size and it is easily scalable with the number of eigenfaces to be computed. The proposed PCNN-based systolic face recognition system has been implemented and evaluated on a Xilinx ML403 evaluation platform with Virtex-4 XC4VFX12 FPGA. The FPGA-based design for a reasonably large-sized face database can process more than 400 faces in a video image frame which is fast enough for video surveillance in busy public places and sensitive locations.**

***Index Terms*—Face recognition, field programmable gate array (FPGA) implementation, principal component neural network, systolic array.**

I. Introduction

FACIAL appearance serves as a natural and effective means for recognition of a person by another. Therefore, researchers have studied face recognition by a computer. While early efforts on automatic recognition were in the 1970s [1], there has been tremendous activity on face recognition (and other biometrics) [2]– [7] during the last decade particularly driven by growing security concerns worldwide.

In an automated video surveillance, it is necessary to detect and recognize faces in the video arriving at a rate of nearly 30 frames/s. Moreover, the recognition process involves comparison of a given face with the enrolled faces in the database. The recognition time depends on the size of the database which may contain thousands of faces (a watch list for example) in security applications at busy places like airports. Since the face recognition process is computation-intensive, dedicated hardware is important.

This paper presents a new algorithm and very large scale integration (VLSI) architecture for automated face recognition. The proposed algorithm is based on principal component analysis (PCA). Principal components correspond to eigenvectors of the data covariance matrix arranged in descending order of eigenvalues. PCA is well suited for face recognition since it allows ignoring common facial features (such as eye, nose, and so on) and, at the same time, extracts the appearance-based dissimilarity in human faces. A direct approach for PCA is based on computation of the entire covariance matrix. However, this requires enormous amount of computation and memory and is especially not suited for VLSI implementation.

A powerful alternative (to direct computation of covariance matrix) for face recognition based on the principal component neural network (PCNN) [8] with generalized Hebbian algorithm (GHA) [9] is proposed in this paper. PCNN with generalized Hebbian learning extracts the principal components (also called *eigenfaces*) directly from the face images in the database. PCNN also has some other advantages. It can be realized by a single layer of linear neurons (without a nonlinear activation function) and its learning algorithm has simple arithmetic operations. Further, inherent features such as parallelism, regularity, and cascadability of the neural network facilitate efficient hardware mapping. Our experiments on the benchmark Yale [10] and FRGC [11] databases with the proposed PCNN-based face recognition system demonstrate a recognition performance more than 85%.

While the PCNN with GHA is generally feasible for VLSI realization, a direct approach based on mapping each neuron

Manuscript received September 23, 2010; accepted January 12, 2011. Date of publication March 28, 2011; date of current version August 3, 2011. This work was supported in part by the Academic Research Fund, under Grant Ref. RG 49/06 of the Ministry of Education, Singapore. This paper was recommended by Associate Editor H. Gharavi.

N. Sudha and A. R. Mohan are with the School of Computer Engineering, Nanyang Technological University, 639798, Singapore (e-mail: sudha@ntu.edu.sg; moha0091@ntu.edu.sg).

P. K. Meher is with the Department of Embedded Systems, Institute for Infocomm Research, 138632, Singapore, and also with the School of Computer Engineering, Nanyang Technical University, 639798, Singapore (e-mail: pkmeher@i2r.a-star.edu.sg).

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

Digital Object Identifier 10.1109/TCSVT.2011.2133210

1051-8215/$26.00 © 2011 IEEE

to a processor is not attractive and feasible in many resource-constrained cases, since it requires several multipliers, and adders for each neuron. A key contribution of this paper is an area-time efficient architecture based on systolic arrays [12]. For straightforward implementation, different steps of computation involved in the algorithm could be mapped on to a set of systolic arrays. However, most of the time only one such systolic array would be in use, while others would remain idle. On the other hand, since the computations of different steps of the algorithm are not identical, it is not straightforward to design an optimized systolic array for the overall computation of the system. One way of optimizing the FPGA implementation of the face recognition system with minimum reconfiguration overhead is to use partial reconfiguration feature of some FPGA devices, where a portion of the design (the reconfigurable components) could be reconfigured while the rest (the static components) remains the same and could be still running during the reconfiguration. However, we find that the reconfiguration time is still too high to have an effective real-time face recognition system. Besides, the partial reconfiguration feature is limited to Xilinx FPGAs, while the FPGAs from other manufacturers do not support this feature and involve still higher reconfiguration time. In this paper, we present an optimized self-configurable systolic architecture which not only could be used for any available FPGA device, but could be used for ASIC as well.

The key strategy used in this paper is to construct fine-grained systolized signal flow graphs (SFGs) for the computation of each individual step of the algorithm, and to map the nodes of the SFG into minimum number of functional units in the processing elements (PEs) by appropriate time-multiplexing of the computations. The proposed face recognition system has the following features.

1) The PCNN-based system is realized on a single linear systolic array consisting of identical PEs with simple and reusable logic.
2) The systolic architecture is designed for both computation of eigenfaces and recognition of a new face. Eigenfaces can be recomputed when the database changes.
3) The proposed design is easily scalable for the number of eigenfaces to be computed. Another interesting feature of the architecture is that it has the flexibility of processing face images and databases of any size.
4) The architecture achieves high-speed due to its small clock period of duration nearly $T_M + 2T_A$, where T_M and T_A are the time required for one multiplication and one addition in a PE.
5) The proposed architecture is self-configurable to have a different data-flow pattern and logic implementation according to the centrally generated mode-control signals.

The rest of this paper is organized as follows. In the next section, prior work on face recognition algorithms and architectures is reviewed. In Section III, the PCNN-based recognition system is described, and its systolic mapping is presented in Section IV. The design of a self-configurable systolic array architecture is described in Section V. FPGA implementation along with comparison with existing hardware implementations of face recognition system are presented in Section VI. Conclusions are given in Section VII.

II. Related Work

We discuss here the prior efforts in face recognition on the algorithmic and architectural fronts.

A. Conventional Approaches for Face Recognition

Eigenface-based recognition [13] by performing PCA is the oldest and popular subspace method for face recognition, where the high-dimensional face image is converted to a low-dimension feature vector. Eigenfaces are found to be generally robust to small translations, rotations and scale changes and can tolerate high blurring effects [14].

Relatively recent subspace methods are based on linear discriminant analysis (LDA) [15] and locality preserving projections (LPP) [16]. LDA is similar to PCA, however, eigenvalue analysis is performed on a *separation matrix* (obtained from the within-class and between-class scatter matrices) in LDA rather than on the data covariance matrix. Both PCA and LDA have been established to be useful under different circumstances. PCA has been shown to be superior to LDA particularly when the number of training samples per class is small. Furthermore, for surveillance type of applications, only one training sample may be available for each subject and here LDA is not suitable since the within-class scatter matrix turns out to be zero matrix. Traditional PCA and LDA perform analysis on vector representation of face images. Of late, 2-D PCA and LDA [17], [18] have been developed for image matrix to improve recognition performance.

The LPP is also an appearance-based recognition method like PCA. For frontal face images, LPP is found to be superior when evaluated on certain benchmark databases. Besides LDA and LPP, face recognition using Bayesian analysis of image differences [19] has also been studied. The authors in [19] defined a probabilistic measure but it turns out that this measure is computationally expensive.

Comparative studies have been performed by many researchers [20]– [24]. These studies show that each method performs better than others in some respects. PCA turns out to be very suitable for security-related applications since only few samples per subject, as in the case of mugshot face database, may be available. To the best of our knowledge, there is no prior work on area-time efficient architectures or efficient FPGA realization for PCA, LDA, LPP, and other methods.

B. Neural Networks for Face Recognition

The neural network-based face recognition approaches include the use of convolutional neural networks [25], radial basis neural networks [26], and other types of neural networks [27]– [29]. All of these focus on recognition performance leading to complex learning algorithms and non-linear neurons. In several of these works, the neural networks act as classifiers. Separate feature extraction algorithms extract relevant features that are fed to the neural network classifiers. The complexity of learning algorithms and feature extraction algorithms make

the existing neural network-based face recognition methods inefficient for hardware mapping. On the other hand, PCNN [8] is a single-layer feed-forward network of linear neurons. It can extract the eigenface features iteratively with a simple Hebbian learning algorithm. It can be therefore efficiently mapped on to a hardware architecture.

C. VLSI Architectures for Face Recognition

Hardware/software co-design for face recognition has been recently explored by researchers. The authors in [30] presented a hardware/software co-design based on template correlation. For the hardware part of the system, a VLSI image correlator has been used in [30]. The preprocessing and post-processing steps are performed in software, where the preprocessing involves the detection and normalization of the face in the image captured by the camera, while the post-processing involves comparing the face with the ones in the database using correlation scores computed by a correlator chip. Since the post-processing time depends on the size of the database and it is performed in software, the system can achieve real-time performance only for moderately-sized databases of roughly 500 images. Recently, two other designs for eigenface-based recognition have been proposed in [31] and [32] where eigenfaces are computed in software and stored in ROM. Besides, the projections of face images in database onto the face space spanned by eigenfaces have also been computed in software. A parallel hardware architecture has been designed only for the recognition of a new face.

In real-world situations, new faces are added and/or some already enrolled faces may be removed from the database quite often. Further due to ageing, changes occur in the face and hence, faces stored have to be replaced by the most recent ones. In view of these, when the face database changes, the eigenfaces and projections have to be recomputed and reloaded manually and it is advantageous therefore to perform eigenface extraction using dedicated hardware. Prior architectural efforts, to the best of our knowledge, have not examined the computation of eigenfaces in custom hardware.

To the best of our knowledge, a simple, linear, and self-configurable systolic array capable of processing 2-D images for face recognition is not available. Our objective in this paper is to present an optimal reconfigurable design which could be realized in any FPGA device as well as in ASIC with significantly faster recognition compared to the FPGA implementation in partially reconfigurable Xilinx devices.

III. Proposed Neural Network-Based Face Recognition System

Before describing the system, in Table I, we put forth the notations which we have used widely in the remainder of this paper.

The proposed face recognition system (Fig. 1) assumes the inputs to be normalized face images from a face detection system which segments and normalizes the facial regions from the images captured by a camera. A single-layer linear feed-forward neural network with generalized Hebbian learning [9], [33] as shown in the figure can extract the eigenfaces of a set S of training faces. In real-world applications, the enrolled faces stored in the database constitute the training set S. S consists of normalized face images. The input $X = [x_1, x_2, \ldots, x_N]$ to the network is a face image in the vectorized form. The number of neurons (or outputs) M determines the number of eigenfaces. The weight vector $W_j = [w_{j1}, w_{j2}, \ldots, w_{jN}]$ associated with neuron j converges to the jth eigenface when trained with faces in S. w_{ji} is the weight associated with the connection from ith input to jth neuron. μ is the learning rate in the range

TABLE I
Notation

S	Set of face images
X_p	Face image in vector form
x_{pi}	ith element of X_p
P	Number of face images
I	Number of training iterations
μ	Learning rate
W_j	Weight vector corresponding to jth neuron
w_{ji}	Weight of the connection between jth neuron and ith input
N	Number of network inputs
M	Number of network outputs
y_{pj}	Output of jth neuron for pth face image input
$\widehat{W}_j$	jth eigenface in vector form
$\widehat{Y}_p$	Projections (in vector form) of X_p onto M eigenfaces
$\hat{y}_{pj}$	jth element of $\widehat{Y}_p$
X^*	New face in vector form
Y^*	Outputs of trained network in vector form when X^* is fed
d_p	l_1 norm of difference between Y^* and $\widehat{Y}_p$
d	Minimum of d_p values
p^*	Identity of new face
PS_{ji}, PD_j	Partial sums
S_k	Select signals of MUXs and DMUXs
M_k	Mode-control signals
n	Number of sub-images
C_T	Number of clock cycles taken for training phase
C_R	Number of clock cycles taken for recognition phase

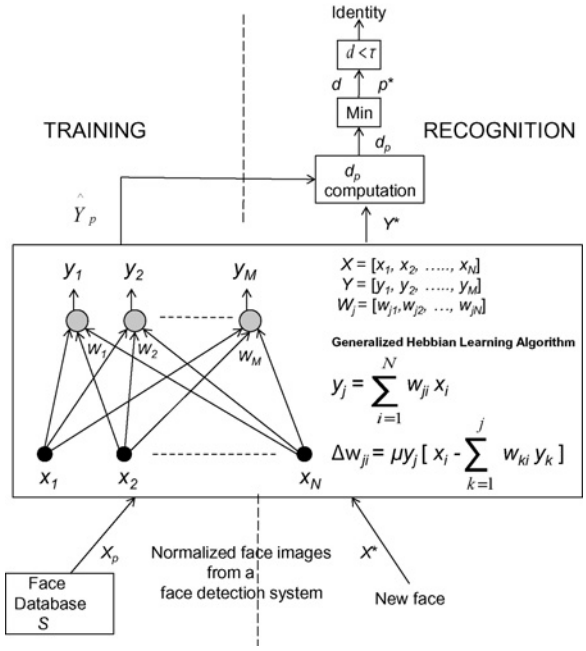

Fig. 1. PCNN-based face recognition system.

[0, 1]. The network is operated in two phases, i.e., 1) training phase, and 2) recognition phase.

1) *Training Phase:* The inputs are face images in training set S. This phase involves the computation of eigenfaces of S and the projection of each face in S on to the space spanned by the eigenfaces. Let S have P face images. That is, $S = [X_1, X_2, \ldots, X_P]$. The network is fed with these face images X_p, for $p = 1$ to P, one after another, while the weights are updated for each input according to the learning rule shown in Fig. 1. The training set of faces is provided as input to the network for several iterations until the convergence of weights, such that after convergence, the maximum change in the weight values remains within a predetermined tolerable limit. The number of iterations can normally be fixed to a larger number I within which the weights converge to eigenfaces $\widehat{W}_j$. Once the network is trained, the network output vector $Y = [y_1, y_2, \ldots, y_M]$ for each face in S is computed and stored. Let $\widehat{Y}_p = [\hat{y}_{p1}, \hat{y}_{p2}, \ldots, \hat{y}_{pM}]$ be the output of trained network for face $X_p = [x_{p1}, x_{p2}, \ldots, x_{pN}]$. The elements of vector $\widehat{Y}_p$ are the projections of X_p on to eigenfaces $\widehat{W}_j$. The recognition of a new face (a normalized face from the detection system) is done using the trained network and the projection vectors $\widehat{Y}_p$, for $p = 1$ to P.

2) *Recognition Phase:* The given new face X^* is fed as input to the trained network and the vector difference between the network output Y^* and each of $\widehat{Y}_p$, for $p = 1$ to P, is computed. The minimum of the l_1 norm values of these difference vectors is given by $d = \min_p(d_p)$ where $d_p = \|Y^* - \widehat{Y}_p\|_1$. If $d < \tau$, where τ is a threshold, the corresponding p is considered as the identity p^* of the new face. Otherwise, the face is classified as "unknown."

For smaller τ, the rejection rate increases and for larger τ, the recognition accuracy decreases. The tradeoff between rejection rate and recognition accuracy will be different for each of the various face recognition applications. In the case of surveillance applications such as security checks at border crossing, the training set consists of mug shots of criminals. τ should be high enough to have the rejection rate close to zero. In the case of authentication applications such as identity check at automated teller machines and entries to restricted areas, τ should be low enough to accept only known faces in the training set.

A. Improvement of Recognition Performance of PCNN-Based System

To obtain a solution that is robust to directional lighting and expression changes and to reduce the size of the network, a modular PCA approach that is based on [35] is performed on sub-images. For this purpose, the face image is divided into smaller equal-sized n sub-images and those sub-images are fed to the network instead of the entire face image. The sub-images for $n = 4$ and $n = 8$ are shown in Fig. 2. Further division of these sub-images results in division of important facial features like eyes. Hence for larger values of n, the global facial features in the image parts are lost and this deteriorates the recognition performance. Two different approaches are used to train the PCNN using sub-images for modular PCA method. One approach uses a single PCNN to process all the sub-images of a given image in a sequence, starting from top-left of the image to the bottom-right of the image. The second approach applies different PCNNs to different parts of the image. Multi-PCNN is expected to perform better in recognition than single-PCNN. However, multi-PCNN consumes more hardware resources compared to single-PCNN. Single-PCNN has been chosen for hardware implementation described later in Section VI. During recognition, the identities of parts of new face are first found out. The identity of the entire face is p^* if the maximum number of its parts is identified as p^*.

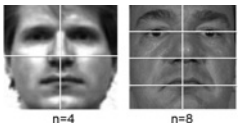

Fig. 2. Sub-images of face images.

Further, in the case of a very large training set, the training set is divided into different subsets and each subset of faces is applied to train one PCNN to improve the performance during recognition. The number of trained PCNNs will be equal to number of subsets. The subset size is constrained by the hardware resources available to implement a PCNN. Samples of same face should be present in the same subset. The face images in a subset are projected onto eigenfaces extracted by the corresponding PCNN. These projections are the outputs of the trained network when the face images are given as input. During recognition, the new face is presented to all PCNNs. The output vector of each PCNN is compared with the projection vectors of faces in its training subset. Each comparison results in a scalar difference value (d_p). The identity of new face is the one that corresponds to the minimum d_p. Besides improving recognition performance, dividing large training set has the advantage of reduced retraining time when there is amendment since only those PCNNs corresponding to the amended subsets are retrained.

The mapping of the system on to a self-configurable systolic architecture is presented in the next section.

IV. Proposed Systolic Mapping of the Neural Algorithm

As discussed in the previous section, the proposed PCNN-based face recognition system operates in two distinct phases, namely, the training phase and the recognition phase. Both these phases are implemented as follows.

1) Steps in training phase.
   a) *T-1*: computation of neuron outputs for a training face X_p.
   b) *T-2*: updating the neuron weights.
   c) *T-3*: computation of projections of X_p on to the eigenfaces.
   Steps T-1 and T-2 are repeated for P training faces during each iteration of training.
2) Steps in recognition phase.
   a) *R-1*: computation of neuron outputs for a new face X^*.
   b) *R-2*: computation of d_p.

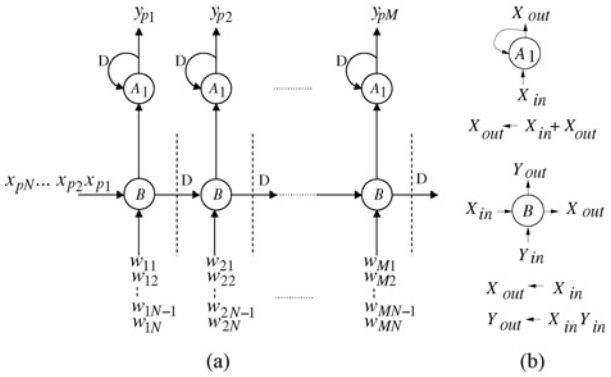

Fig. 3. (a) SFG for computation of neuron outputs for a training face. (b) Function of node-A_1 and node-B.

We construct here the fine-grained SFG of the computation involved in each of these steps and map all the SFGs corresponding to different computational steps into a single systolic array architecture.

A. Computation of Neuron Outputs for Training Faces

The neuron outputs $y_{pj}, 1 \le j \le M$, for an input face $X_p = [x_{p1}, x_{p2}, \ldots, x_{pN}]$ in the vectorized form are computed according to GHA as a set of inner-products, given by $y_{pj} = \sum_{i=1}^{N} w_{ji} x_{pi}$ where w_{ji}, for $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, M$ are the weights connecting the ith input to the jth neuron. The index p refers to the training face, for $1 \le p \le P$. A fine-grained SFG depicting all the inputs, outputs, computations and dependencies is shown in Fig. 3(a). It consists of N pairs of nodes, where each pair consists of a node-A_1 and a node-B. The function of nodes A_1 and B are shown in Fig. 3(b). The inputs to the node-B include the gray values $x_{pi}, 1 \le i \le N$, and the weight values as shown in the figure. It multiplies the input gray value with the corresponding weight, and passes the product value to node-A_1. Node-A_1 accumulates the input available from node-B. A pair of nodes (B and A_1) of the SFG thus performs a multiply-accumulate operations to compute a value of y_{pj} in N consecutive time-steps, such that y_{pj} for $j = 1, 2, \ldots, M$ are computed in M different SFG nodes.

B. Updating Weights

Once the neuron outputs are computed, the weights are updated. The new weight values of the neurons are computed according to the following equation:

$$\begin{aligned} (w_{ji})_{\text{new}} &= (w_{ji})_{\text{old}} + \Delta w_{ji} \\ \Delta w_{ji} &= \mu y_{pj}[x_{pi} - \sum_{k=1}^{j} w_{ki} y_{pk}] \end{aligned} \quad (1)$$

for $1 \le j \le M$, $1 \le i \le N$, and $1 \le p \le P$. The updating of weights is done in two stages. In the first stage, the summations in (1) are performed as

$$PS_{ji} = \sum_{k=1}^{j} w_{ki} y_{pk} = PS_{(j-1)i} + w_{ji} y_{pj} \quad (2)$$

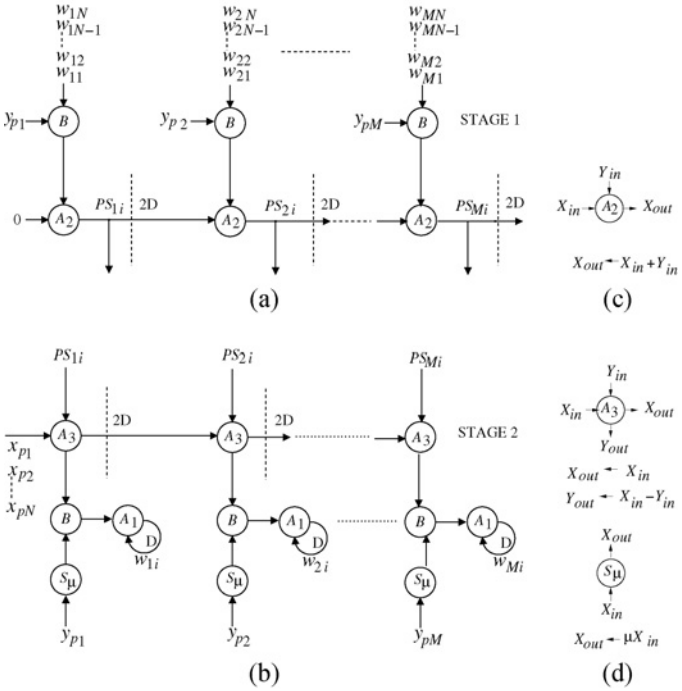

Fig. 4. SFGs for weights updating. (a) SFG for the first stage given by (2). (b) SFG for the second stage given by (3). (c) Function of node-A_2. (d) Function of node-A_3 and node-S_μ.

for $1 \le j \le M$, $1 \le i \le N$, and $1 \le p \le P$. In the second stage, the new values of weights are computed using PS_{ji} as

$$(w_{ji})_{\text{new}} = (w_{ji})_{\text{old}} + \mu y_{pj}[x_{pi} - PS_{ji}] \quad (3)$$

for $1 \le j \le M$, $1 \le i \le N$, and $1 \le p \le P$.

A fine-grained SFG for both of these stages of weight updating is shown in Fig. 4. The function of node-B is the same as that described in Fig. 3(c). The functions of nodes A_2 and A_3 are shown in Fig. 4(c) and (d), respectively. For the first stage of weight updating [according to (2)], the jth node-B is fed with the neuron output y_{pj} which is available after the computation of the previous step. The node multiplies y_{pj} with a corresponding weight value and passes the product value to node-A_2. Node-A_2 adds the product received from node-B with the partial sum received from its predecessor node as horizontal input X_{in}. The first node-A_2 of the SFG is fed with "0" as horizontal input X_{in} since it has no previous node. The new partial sum computed by node-A_2 is passed to the next node as well as to node-A_3 for the computation of the second stage of weight updating [according to (3)]. The pixel values $x_{pi}, 1 \le i \le N$, are fed sequentially to node-A_3 as the other input. This node computes the difference $[x_{pi} - PS_{ji}]$ and passes it to node-B. The node-B multiplies the input received (from node-A_3) with the scaled input y_{pj} . The scaling is done by node-S_μ. The resulting product from the jth node-B is Δw_{ji} which is passed to the jth node-A_1 for updating w_{ji}. Since the updating is done in two stages and each stage requires one clock cycle for its processing in hardware, a delay of 2-D is introduced in the SFGs for pipelining.

C. Computation of Projections of Training Faces onto Eigenfaces

After repeated updating of weights for different input faces X_p in several iterations, the weights of the trained network converge to optimized eigenfaces $\widehat{W}_j$. The projection of each face X_p onto $\widehat{W}_j$ is computed as follows:

$$\hat{y}_{pj} = \sum_{i=1}^{N} \hat{w}_{ji} x_{pi}. \tag{4}$$

The SFG depicting the computation is similar to Fig. 3(a). Optimized weight values $\hat{w}_{ji}$ replace w_{ji} in the figure and the outputs are the projections $\hat{y}_{pj}$ of training faces onto eigenfaces given by the optimized weights.

D. Computation of Neuron Outputs for a New Face

Given a new face $X^* = [x_1^*, x_2^*, \ldots, x_N^*]$, the projections of X^* onto eigenfaces $\widehat{W}_j$ are computed as

$$y_j^* = \sum_{i=1}^{N} \hat{w}_{ji} x_i^*. \tag{5}$$

The SFG depicting the computation is the same as Fig. 3(a), except that the inputs in this case are the pixel values of new face and the weights are the optimized ones.

E. Computation of Difference Between Projections

d_p is computed as the sum of differences between the corresponding components of the projected outputs $Y^* = [y_1^*, y_2^*, \ldots, y_M^*]$ of a new face X^* and $\widehat{Y}_p = [\hat{y}_{p1}, \hat{y}_{p2}, \ldots, \hat{y}_{pM}]$ of each of the P face images in set S. The computations of $\widehat{Y}_p$ and Y^* are given by (4) and (5), respectively. d_p is computed for each face image p according to the equation

$$d_p = \sum_{j=1}^{M} |\hat{y}_{pj} - y_j^*| \quad \text{for} \quad 1 \le p \le P.$$

The fine-grained SFG depicting the computation of d_p is shown in Fig. 5(a). The function of node-A_4 is shown in Fig. 5(b), while that of node-A_2 is described in Fig. 4(c). Node-A_4 is fed with $\hat{y}_{pj}$ and y_j^* which computes the absolute difference between them. The difference value is then passed to node-A_2 as vertical input which adds it to the accumulated differences received from the previous node-A_2 as horizontal input. The jth node-A_2 thus computes the partial sum $PD_j = \sum_{k=1}^{j} |\hat{y}_{pk} - y_k^*|$. The horizontal input X_{in} to the first node-A_2 is "0." d_1 is the first output of the Mth node A_2 (the right-most node A_2) after M time-steps. In the successive $M-1$ time-steps, d_2 to d_M are available as output from the same node A_2.

V. Proposed Systolic Architecture

In this section, we analyze the computations associated with the SFGs of different steps of processing in the proposed PCNN-based face recognition system, and formulate a mapping of all the SFGs to a single systolic array architecture.

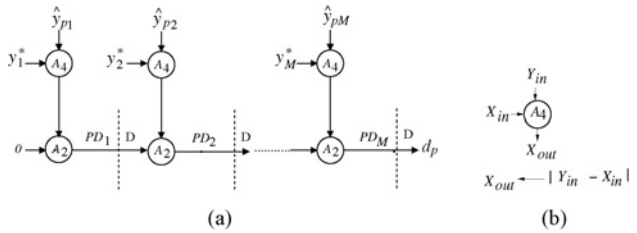

Fig. 5. (a) SFG for the computation of d_p. (b) Function of node-A_4.

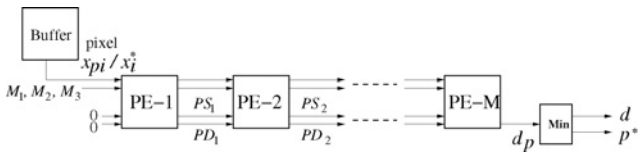

Fig. 6. Systolic architecture. M_1, M_2, and M_3 are mode-control signals corresponding to Mode-1, Mode-2, and Mode-3, respectively. PS_j and PD_j are the partial sums computed by the jth PE in Mode-2 and Mode-3, respectively.

The computations shown in the SFGs for the steps T-1, T-3, and R-1 are identical. All the five steps of the training and recognition phases therefore could be classified into three modes of operations, namely:

1) *Mode-1*: computation of neuron outputs;

2) *Mode-2*: updating weights;

3) *Mode-3*: computation of d_p.

In a straightforward implementation, the SFGs for the different modes could be mapped onto different systolic arrays each consisting of M processing elements (PEs) by feed-forward cut-set retiming. For better hardware utilization without significantly degrading the timing performance the computation of all the five steps could be multiplexed into a single systolic array of M PEs as shown in Fig. 6. M determines the number of eigenfaces extracted by the array.

The overall face recognition system, however, could be obtained by interfacing appropriate inputs and outputs to the PEs at different modes of operation. The presentation of mode-control signals determines whether the system is in training or in recognition phase. When the system is in training phase, Mode-1 and Mode-2 are repeated for training with the input faces. In recognition phase, the system is put in Mode-1 followed by Mode-3. The face images are stored in a buffer from which they are sent (pixel by pixel) to the systolic array. The output d_p of the array during the recognition phase is given to a minimum finding logic to determine d, the minimum of d_p values, and its corresponding identity p^*.

A. Processing Element Design

For the implementation of operations of Mode-1 (given by SFGs in Fig. 3), each PE needs one multiplier, one adder and a register for output accumulation. For the implementation of Mode-2 operations (given by SFG in Fig. 4), two multiplications, three additions and one scaling operation are required. This is implemented by a PE consisting of one multiplier, two

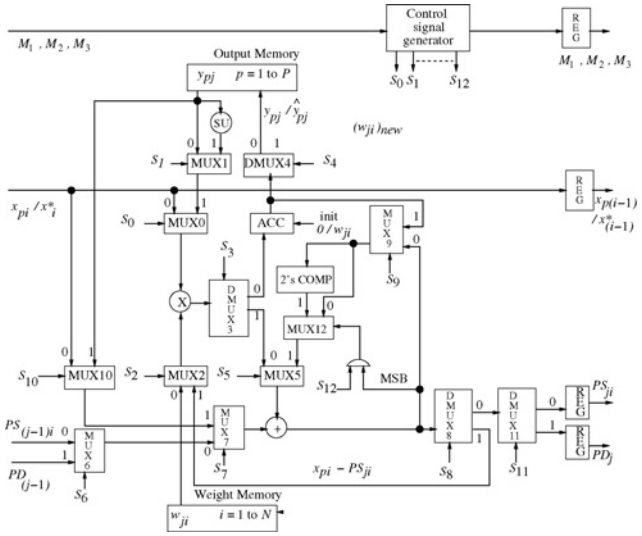

Fig. 7. Processing element architecture. The structure and function refers to the jth PE. SU indicates a scaling unit to perform multiplication with the learning rate μ.

adders and one hardwired scaling unit, if we assume the cycle time to be of duration $T = T_M + 2T_A$, where T_M and T_A are the time required for one multiplication and one addition, provided that the SFG is implemented in two cycles and the scaling is performed with the multiplications in parallel during the first cycle. The implementation of Mode-3 operations (given by SFG in Fig. 5) is relatively simpler. It involves only two additions. When these additions are performed in two different half cycles, only one adder is sufficient to perform Mode-3. Therefore, the arithmetic units in a PE are an adder, multiplier, accumulator and a scaling unit.

The proposed structure of the PE with reusable arithmetic units is shown in Fig. 7. The weight memory is of size N to store the eigenface extracted by that PE, while the output memory is of size P to store the P output values y_{pj}, for $1 \le p \le P$. The data flow and computations during different phases and the generation of control signals are explained in detail next.

1) *Data Flow and Computations During Training Phase:* In Step T-1 (given by SFG in Fig. 3), x_{pi} is fed to the PE. The PE is in Mode-1 for N inputs, $x_{pi}, i = 1$ to N, of a training image. The PE accumulates the products $x_{pi}w_{ji}$. The final output y_{pj} of the accumulator is stored in the memory.

Step T-2 is executed in two cycles. The PE is in Mode-2. x_{pi} is fed to the PE and stays at the input line for both cycles. In the first cycle, as given by the SFG in Fig. 4(a), y_{pj} is fed to the multiplier and PS_{ji} is computed when the control signals are assigned appropriate values. PS_{ji} is passed to the next PE. In the second cycle, the output PS_{ji} is negated by two's complement operation and fed back to the adder. The negated PS_{ji} is added with input x_{pi}. The output of the adder, $x_{pi} - PS_{ji}$, is multiplied with the scaled y_{pj}, i.e. μy_{pj}. The product $\mu y_{pj}[x_{pi} - PS_{ji}]$ is added to the weight w_{ji} stored in the accumulator. The output of the accumulator is the new weight which is stored in the weight memory. Note that the operation in the Step T-3 is same as that of Step T-1.

TABLE II
CONTROL SIGNAL GENERATION

Signal	Value	Condition
S_0	0	Mode-1
	1	Mode-2
S_1	0	Mode-2 and Cycle-1
	1	Mode-2 and Cycle-2
S_2	0	Mode-1 or (Mode-2 and Cycle-1)
	1	Mode-2 and Cycle-2
S_3	0	Mode-1 or (Mode-2 and Cycle-2)
	1	Mode-2 and Cycle-1
S_4	0	Mode-1
	1	Mode-2 and Cycle-2
S_5	0	Mode-2 and Cycle-1
	1	(Mode-2 and Cycle-2) or Mode-3
S_6	0	Mode-2 and Cycle-1
	1	Mode-3
S_7	0	(Mode-2 and Cycle-1) or (Mode-3 and HalfCycle-2)
	1	(Mode-2 and Cycle-2) or (Mode-3 and HalfCycle-1)
S_8	0	(Mode-2 and Cycle-1) or (Mode-3 and HalfCycle-2)
	1	Mode-2 and Cycle-2
S_9	0	Mode-2 and Cycle-2
	1	Mode-3
S_{10}	0	Mode-2 and Cycle-2
	1	Mode-3
S_{11}	0	Mode-2 and Cycle-1
	1	Mode-3 and HalfCycle-2
S_{12}	1	Mode-3 and HalfCycle-2

2) *Data Flow and Computations During Recognition Phase:* In Step R-1, the data flow in PE is the same as Step T-1. x_i^* is fed to the PE. y_j^* is computed using the trained weights, $w_{ji}, i = 1$ to N, available from the memory of the PE and the inputs $x_i^*, i = 1$ to N, of the new face.

Step R-2 as described by the SFG in Fig. 5 is executed by using the adder in both half cycles of a cycle. The PE is in Mode-3. In the first half cycle of Step R-2, y_j^* stored in the accumulator is negated and fed to the adder. The second input to the adder, $\hat{y}_{pj}$, is taken from the memory. In the second half cycle, the output of adder is fed back to itself after taking the absolute value of it. This is done by testing the MSB of the output. This absolute value is added to PD_{j-1}. The output PD_j is fed to the next PE.

3) *Generation of Control Signals:* The select signals (S_0 to S_{12}) of the multiplexers and demultiplexers control the flow of data to the arithmetic units in different modes of operations. These control signals can be generated from the mode-control signals and the clock signal as in Table II. Cycle-1 and Cycle-2 are alternate cycles. HalfCycle-1 and HalfCycle-2 refer to the first and the second halves of a cycle.

The sequence of activation of mode-control signals and the presentation of inputs during the training and recognition phases in the jth PE is given by the flowcharts in Fig. 8. The values of p and iteration numbers are generated by external counters. During the training phase, the architecture is input with training faces X_p when it is allowed to run in Mode-1 and Mode-2 alternatively. The PE is in Mode-1 for N cycles and in Mode-2 for $2N$ cycles. The training phase results in the computation of projection vector elements $\hat{y}_{pj}$. In the recognition phase, the PE is input with new face X^*. It is in Mode-1 for N cycles to compute y_j^*. Then the PE is put in

Mode-3 for P cycles. The stored $\hat{y}_{pj}$ along with the computed y_j^* are used when the PE is in Mode-3 to compute the partial sum PD_j for the d_p values. It is worth noting that all PEs have the same sequence of operations. However, they need not be in the same mode of operation at any time.

The PE is *self-configurable* in run time. It changes its functionalities depending on the mode of operation. To achieve such self-configurability, the data-flow pattern in the PE is reconfigured according to the mode of operation to perform the desired function. The reconfiguration is achieved by using multiplexers and demultiplexers which change the data-flow by appropriately generating the select signals (derived from the mode-control signals). The arithmetic units in the PE are reused in different modes of operation.

B. Operation of Overall Architecture

During each cycle, the systolic architecture (shown in Fig. 6) is fed with the mode control signals and a pixel value. These inputs are passed on to the successive PEs at successive clock cycles along with the partial sum PS_{ji} or PD_j if computed. The scheduling of different steps in the training and the recognition phases for the overall systolic architecture is shown in Fig. 9. Note that different PEs may be functioning in different modes of operation at a given cycle in the pipelined structure. The overlapping of different computations at certain time intervals [for example, between Nth and $(N + M - 1)$th clock cycles in Fig. 9(a)] indicates the PEs in different modes. Mode-1 takes $N+M-1$ cycles to process a face image vector with N elements. Mode-2 and Mode-3 take $2(N+M-1)$ and $(P+M-1)$ cycles, respectively. External counters are used to count the pixels input to the system and to set the mode-control signal inputs to the array at appropriate time specified in Table III. The presentation of data for different mode-control inputs is also shown in the table. During training phase, M_1 is set for the first N clock cycles followed by setting M_2 for the next $2N$ cycles. The input data will be the pixels x_{pi}, $1 \leq i \leq N$ of the first training image ($p = 1$). When M_2 is set, each pixel is input for two clock cycles. After a delay of $M - 1$ cycles, M_1 and M_2 are set again for the same number of cycles and the pixels of second training image ($p = 2$) are fed as before. The pattern of setting these two mode control signals and the presentation of image pixels is repeated IP times. This performs the computation of neuron outputs and weights updating. The projections are then computed by the architecture by setting M_1 for PN cycles. During recognition phase, M_1 is set for N cycles followed by M_3 for P cycles. When M_1 is set, pixels x_i^*, $1 \leq i \leq N$, of new face image are fed to the array one after another. When M_3 is set, the computation of d_p starts. However the output d_1 is available from the Mth PE after $M - 1$ cycles. The remaining outputs d_2 to d_P are available in the successive clock cycles.

1) *Computational Analysis:* We give here a detailed computational analysis for the proposed architecture assuming that a total of M PEs is employed. The overall results can be summarized as follows and the details are given via Propositions 1, 2, 3 and 4 (and Remark 1).

a) The entire training phase takes C_T clock cycles (Propositions 1 and 2), where $C_T = IP(3N+M-1)+PN+(M-1)$.

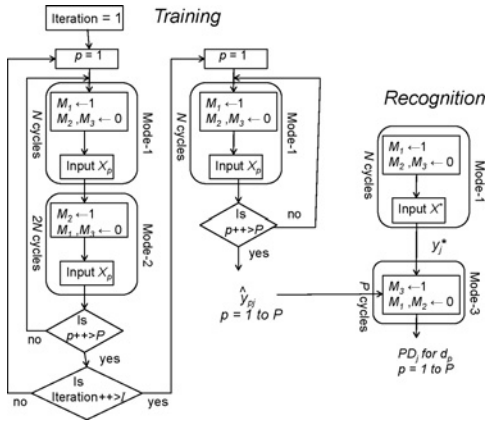

Fig. 8. Combined control flow diagram for the jth PE in training and recognition phases.

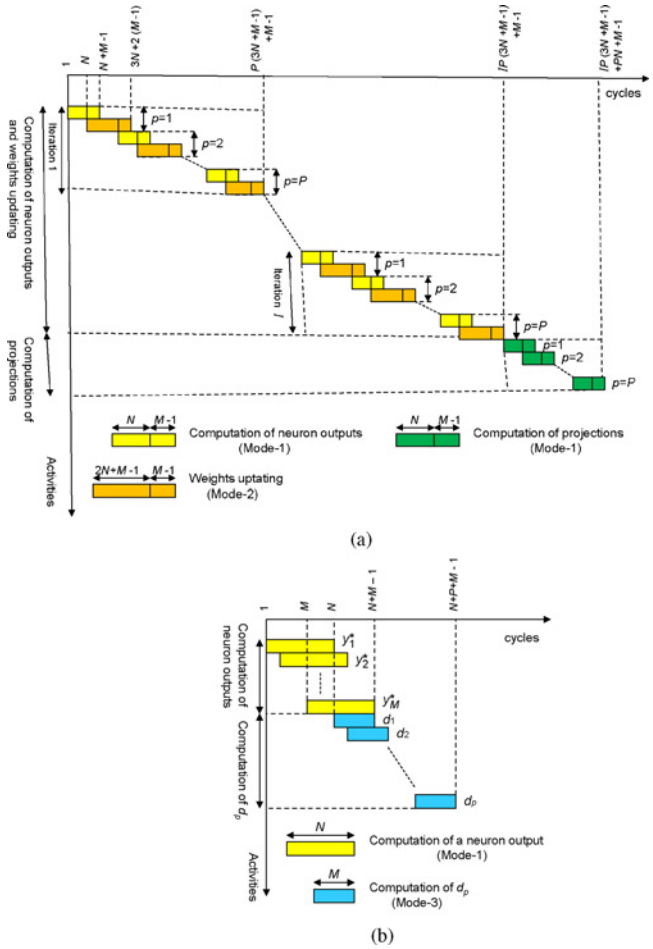

Fig. 9. Scheduling of computations. (a) Scheduling during training phase. (b) Scheduling during recognition phase.

b) The recognition phase takes C_R clock cycles (Propositions 3 and 4), where $C_R = N + P + M - 1$.

Proposition 1: The computation of neuron outputs and weight updating takes $(IP(3N + M - 1) + M - 1)$ cycles.

Proof: Each of I iterations during the training phase involves operations of Mode-1 followed by Mode-2 for each of the P faces to be trained as shown in Fig. 9(a). In the first

TABLE III
ACTIVATION OF MODE-CONTROL SIGNALS AND PRESENTATION OF INPUT DATA

Phase	Mode-Control Signal	Starting Clock Cycle	Ending Clock Cycle	Input Data
Training	M_1	$l(3N+M-1)+1$ $0 \le l < IP$	$l(3N+M-1)+N$ $0 \le l < IP$	$[x_{p1}, x_{p2}, \ldots, x_{pN}]$, $1 \le p \le P$ for I times
		$IP(3N+M-1)+1$	$IP(3N+M-1)+PN$	$[x_{p1}, x_{p2}, \ldots, x_{pN}]$, $1 \le p \le P$
	M_2	$l(3N+M-1)+N+1$ $0 \le l < IP$	$l(3N+M-1)+3N$ $0 \le l < IP$	$[x_{p1}, x_{p1}, x_{p2}, x_{p2}, \ldots, x_{pN}, x_{pN}]$ $1 \le p \le P$ for I times
Recognition	M_1	1	N	$[x_1^*, x_2^*, \ldots, x_N^*]$
	M_3	N+1	N+P	–

iteration, when the first face image is fed, the computation of Mode-1 is started in the first cycle and continues till the $(N+M-1)$th cycle. The execution of Mode-2 then starts at Nth cycle and continues till cycle $(3N+2M-2)$. For the second face, the computation starts at $(3N+M-1)$th cycle and continues till cycle labeled $2(3N+M-1)+M-1$. The first iteration for the Pth face image therefore starts at $((P-1)(3N+M-1))$th cycle and ends at $(P(3N+M-1)+M-1)$th cycle. The computations for other iterations are scheduled similarly one after the next so that all the I iterations of training for all the P face images are completed in the $(IP(3N+M-1)+M-1)$th cycle. ■

Proposition 2: The computation of projections takes $(PN+M-1)$ cycles.

Proof: Once the training for all the face images is over at the first PE at the $(IP(3N+M-1))$th cycle, the computation of step T-3 starts for the computation of the projections. The computation of the projection for the first face image is performed for $(N+M-1)$ cycles, but that of the second face can start after N cycles, since the first PE completes its computation for the first image in N cycles. The computation of the projection for the Pth face starts after $(P-1)N$ cycles and continues for the next $(N+M-1)$. The computation of projections starts at $(IP(3N+M-1))$th cycle and gets completed at $(IP(3N+M-1)+PN+M-1)$th cycle. ■

We now present the results pertaining to recognition phase. Please refer Fig. 9(b) for the following propositions.

Proposition 3: The computation of neuron outputs for a new face takes $(N+M-1)$ cycles.

Proof: Let us assume that the computation of neuron outputs for a new face during the recognition phase starts at the first PE in the first cycle. It continues in the first PE up to Nth cycle. In the second PE, it starts from the second cycle and continues for the next N cycles. In the Mth PE, it continues from Mth cycle to $(N+M-1)$th cycle. ■

Proposition 4: The computation of the projection differences takes $(P+M-1)$ cycles.

Proof: The computation of projection difference d_1 starts at the first PE at the Nth cycle and gets completed in $(N+M-1)$th cycle (in the Mth PE). The computation of d_2 starts in the $(N+1)$th cycle and gets completed in $(N+M)$th cycle, while the computation of d_p starts at $(N+P-1)$th cycle and gets completed in $(N+M+P-1)$th cycle. ■

Remark 1: It may be noted that there is an overlap of $M-1$ cycles between the computation of projections (Proposition 2), and the computation of neuron outputs and weight updating (Proposition 1) in the training phase. Similarly, there is an overlap of $M-1$ cycles between the computation of neuron outputs (Proposition 3) and the computation of projection differences (Proposition 4) in the recognition phase. Hence, the total number of clock cycles taken in each phase is not given by the summation of cycles for the two operations.

Remark 2: The benefits of a systolic realization are seen from the low hardware complexity and high operating frequency. Each PE is devoid of arithmetic units that are not amenable for a hardware-friendly solution. The cycle time is small due to simple computational units.

In the case of a single-PCNN based modular PCA method, the architecture is trained with all sub-images. Hence P is replaced by nP. The number of clock cycles taken by the training phase in this case, denoted by C_T^s, is given by

$$C_T^s = nIP(3N+M-1) + nPN + (M-1). \tag{6}$$

During recognition, each sub-image of the new face is compared with the corresponding sub-images of P training faces. Since there are n sub-images, the number of clock cycles for recognition phase, denoted by C_R^s, is given by

$$C_R^s = n(N+P+M-1). \tag{7}$$

VI. IMPLEMENTATION, PERFORMANCE STUDIES AND COMPARISONS

A. Hardware Implementation

The proposed face recognition system has been implemented on a Xilinx ML403 evaluation platform for embedded systems development. The platform is equipped with a Xilinx Virtex-4 XC4VFX12 FPGA, 64MB DDR SDRAM and industry-standard peripheral connectors and interfaces.

1) *Implementation on FPGA:* The proposed systolic architecture for the single-PCNN case has been coded in Verilog in behavioral style. From Fig. 12, it is clear that there is no significant change in recognition rate when the number of neurons exceeds 16. Therefore, the number of PEs has been taken to be 16 although the proposed architecture is easily scalable for any number of PEs. The size of the registers has been taken to be 16 bits which is double the size of input pixel value (to avoid overflow in the result of multiplication). All the results of computations have been represented as integer quantities. In the case of $n = 8$, an image of size 32×32 is divided into eight parts, each of size 8×16. Hence, N has been taken to be 128 in the design. Memory elements have been used to store the weights corresponding to $N = 128$ and also the projected outputs of P training images.

TABLE IV
RESOURCE UTILIZATION FOR THE FPGA DESIGN FOR THE TRAINING SET OF 1024 FACES

Component	Available	Usage	
Slices	5472	1420	(25%)
Slice flip flops	10944	888	(8%)
Four input LUTs	10944	2384	(21%)
Bonded IOBs	240	25	(10%)
18Kb block RAMs	36	32	(88%)
GCLKs	32	2	(6%)

The designs for different values of P in powers of two have been then implemented on the target device XC4VFX12 using Xilinx ISE 10.1. The target device has a package number of SF363 and a speed grade of -12. Maximum usage of device occurred at $P = 1024$ and the implementation results in this case are given in Table IV. For all other lower values of P, the usage of components is nearly the same due to fixed number of PEs. The number of block RAMs (BRAMs) consumed by the designs is 32 since the design synthesizes 16 PEs and each PE has two memory units. However, when P increases, there is a significant rise in the actual usage of block RAMs synthesizing the output memory unit. Maximum usage of these block RAMs occurred for $P = 1024$. When the training set is larger, there is a need to divide the training set into different subsets so that the FPGA implementation is constrained by the subset's size instead of the size of the entire training set.

The maximum usable frequency of operation is 136.243 MHz. This remains constant for the designs for any value of P because the clock frequency is determined by the maximum signal propagation delay in the combinational logic path.

2) *Experimental Setup for Testing Design:* The netlist for $P = 1024$ has been downloaded to FPGA on the evaluation platform. Data was sent from PC to the ML403 board using HyperTerminal running on Windows XP. Xilinx's IP core UARTLite was instantiated to receive data through the RS232 port. The data received at universal asynchronous receiver/transmitter (UART) buffer was then transferred to DDR SDRAM by an interfacing program (in C) running on PowerPC (hardcore processor in the FPGA device). The PowerPC-based embedded system was developed using Xilinx Platform Studio to interface the FPGA design with LCD and RAM in the board through processor local buses.

The overall experimental setup is shown in Fig. 10. The SDRAM receives face images from PC and sends the data (pixel by pixel) to the FPGA during training and recognition phases. During training, the computation of eigenfaces and projections of training faces onto eigenfaces were performed on the FPGA. These computed quantities were then used for recognition. It is worth noting that the entire training set (of faces) is not stored in the FPGA. During recognition phase, the identity of new face computed by FPGA phase was displayed on the LCD. "0" was displayed if the identity was not known.

The FPGA in our experimental setup acts as a co-processor executing the computation-intensive face recognition task of a complete system with face detection and recognition. In the experimental setup, it is assumed that a camera is connected to a PC, and the PC runs the face detection program to segment

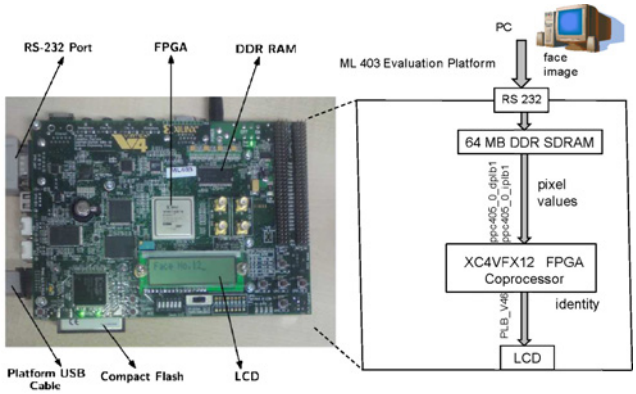

Fig. 10. ML403 board components and connections relevant to our experiment.

facial regions in the captured video images and normalize them. These normalized face images are either added to the training set for training or directly sent to FPGA co-processor for recognition. However, if the face detection is performed by a dedicated processor, then the complete system could be integrated into a smart surveillance camera.

3) *Handling Large Training Set Size by Run-Time Configuration of BRAMs on FPGA:* The FPGA implementation has been extended to handle training sets with more than 1024 images. The training set is divided into subsets of size 1024 or less. The number of PCNNs is equal to the number of subsets as described in Section III-A in the case of single-PCNN. However, the systolic architecture is the same for all PCNNs and only the contents of weights and output memory units vary. Since these memories are mapped onto BRAMs in FPGA, these BRAMs are only reconfigured during training and recognition phases. The weights and projections of the subsets are stored in the external RAM. When a face is added or removed, these values of the amended subset are recalculated. The systolic architecture implementation in FPGA is put in training phase and it is fed with the faces of that subset. After training, the BRAM contents giving the new weight and projection values are transferred to the external RAM. Hence, the total training time (T_T) involves the data transfer time (t_1) and the training time of the systolic array (t_T). That is, $T_T = t_T + t_1$. Since T_T depends on the subset's size and not on the entire training set's size (P), T_T is constant for any P when the subset size is fixed. During recognition, the BRAMs are loaded with the weights and projections of each subset at a time and the new face is fed to the systolic array implementation on FPGA. If there are s subsets, then the total recognition time $T_R = s(t_R + t_2)$ where t_2 is the loading time for BRAMs and t_R is the recognition time of systolic array.

A C program for the data transfer between BRAMs and SDRAM has been run on PowerPC in order to estimate t_1 and t_2. For the subset size of 1024 and for the single-PCNN design implemented on FPGA (Section VI-A1), t_1 and t_2 are estimated to be 0.025 s. t_T in the case of 1024 faces is 2.4 s (see Fig. 14). Hence $T_T = 2.425$ s. The recognition performance of FPGA-based system for the training set size (P) larger than 1024 is shown in Table V. For P as large as 10000 images, the FPGA-based recognition system with an

TABLE V
RECOGNITION PERFORMANCE OF FPGA-BASED FACE RECOGNITION SYSTEM WITH EXTERNAL RAM FOR LARGE TRAINING SETS

Training set size (P)	2048	4096	8192	16 384
Faces recognized per second	19	9	4	2

Fig. 11. Sample images of faces of a person in the Yale (first four) and FRGC (last four) databases. The first two FRGC samples are controlled images and the other two are uncontrolled images.

external RAM can recognize around two faces per second. The recognition system is still suitable for surveillance at narrow or restricted entrances through which only one person can pass. The performance could be enhanced for surveillance at busy places by implementing the design on a more powerful FPGA device with more BRAMs.

B. Performance Studies

The performance of FPGA-based system has been evaluated on Yale [10] and FRGC [11], [34] face databases.

1) *Database Description:* The Yale database has normalized face images of 15 persons, each with 11 samples. The images vary widely in illumination and expression. Fig. 11 shows samples of one person.

The FRGC database is a large-scale database consisting of face images captured at controlled and uncontrolled environment. The database has frontal smiling and neutral faces. Four controlled and two uncontrolled images were collected from each person in a session. FRGC ver2.0 provides data sets for six experiments. We have considered experiment 4 whose data set consists of 12 776 face images from 222 people for the purpose of training and 24 042 face images from 466 people for validation purpose. The FRGC also provides files containing the locations of eyes, nose, and mouth. The face region in each image was segmented using this information. Fig. 11 shows samples of segmented faces of a person.

2) *Performance Evaluation:* In our experiments, five samples of each person in the Yale database have been used for training, and the rest have been used for testing. Altogether, 75 images form the training set and 90 images form the test set. In the case of FRGC, four training images (two controlled and two uncontrolled) of each subject have been considered for training. For recognition studies, four images (two controlled and two uncontrolled) of each subject from the validation set of Experiment 4 have been considered. Hence, the training and test sets have 888 and 1864 images, respectively. The images were converted to gray-scale images. All the images used for experiments have been resized to 32×32. We have also evaluated the performance of multi-PCNN approach on an experimental setup that configures BRAMs for implementing different PCNNs as described in Section VI-A3.

The results of modular PCA approach are plotted in Fig. 12 for both databases. The figures show the plots of recognition rate for varying number of neurons M and the normalized τ set to 0.35 (determined empirically based on inter-class and intra-class d_p values). The plots are for different number of sub-images n. The recognition rate is defined as the ratio of the number of images that were recognized correctly to the number of the images in the test set. It can be observed that the recognition rate increases as M increases and it reaches saturation for a particular value of M, say M_n for each n. There is no significant improvement in performance for $M > M_n$. M_n is close to 16 in both cases. It can be seen that best performance in the case of Yale and FRGC databases is achieved for $n = 4$ and $n = 8$, respectively. The best values of n differ in both databases because Yale database faces have wide expression variations resulting in significant local distortion for large n. It can also be observed from Fig. 12 that multi-PCNN shows improvement in recognition performance over single-PCNN especially in the case of Yale database.

The performance of division of training set into different subsets as proposed in Section III-A is also evaluated on FRGC data set. M and n are set to 16 and 8, respectively. Experiments are conducted for varying subset sizes. In the case of each subset size, different PCNNs are trained with different subsets of training images. The test faces are then given to these trained PCNNs to compute the recognition rate. The experimental setup on ML403 board applies run-time configuration of BRAMs to implement different PCNNs as described in Section VI-A3. From Fig. 13, it can be observed that the recognition performance improves for smaller subset sizes.

For the purpose of comparison, the neural network-based face recognition system has been implemented in C++ running on a PC with Intel Core2 Duo E8500 processor, 3.16 GHz clock, 8 GB physical memory, and Windows 7 operating system. In the case of FRGC data set which is larger, the time taken for training for 100 iterations using single-PCNN with $M = 16$ and $n = 8$ is 3.46 s while the average time taken to recognize a test image is 0.0465 s. This results in the recognition of only 21 faces per second. The recognition time is too large for real-time video surveillance applications in which images normally arrive at a rate of 30 frames/s. Moreover, more than one face may have to be recognized in a frame. In the case of parameter settings $I = 100$, $N = 128$, $M = 16$, $P = 888$, and $n = 8$, the training and recognition times on FPGA are computed to be 2.09 s and 60 μs, respectively, using the frequency of operation (136.243 MHz obtained from FPGA implementation of design) and the number of clock cycles (C_T^s and C_R^s) required. This implies that the proposed FPGA-based face recognition system can recognize up to 16 518 faces per second. For a video stream of 30 frames/s, up to 550 faces can be recognized in a frame. The graph showing the recognition performance for different training set size (P) is given in Fig. 14. It is observed that the FPGA-based recognition system can recognize more than 400 faces per frame for a reasonably large training set. The system is therefore suitable for real-time surveillance in busy public places.

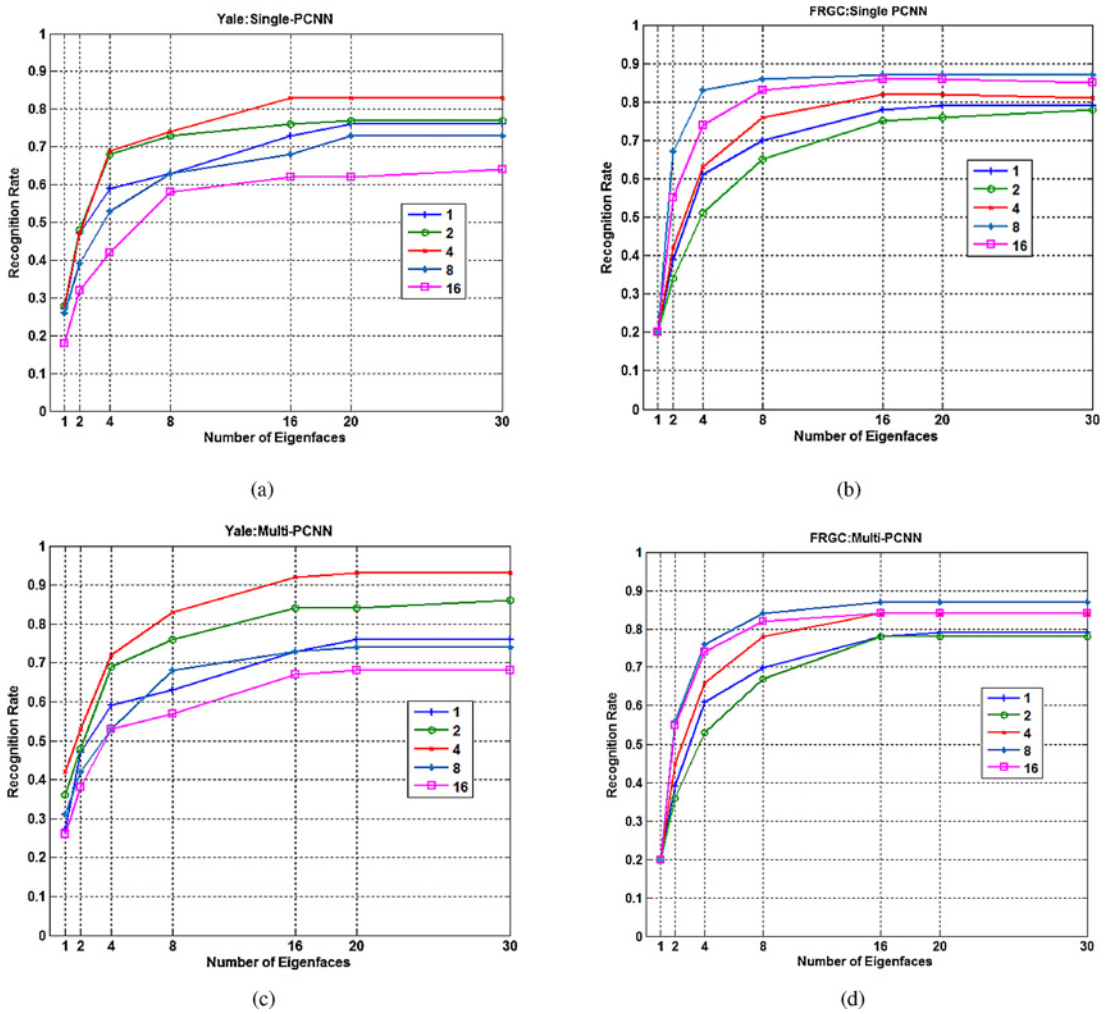

Fig. 12. Performance results of implementation. (a) Single-PCNN for Yale database. (b) Single-PCNN for FRGC database. (c) Multi-PCNN for Yale database. (d) Multi-PCNN for FRGC database. The numbers in the legend are the number of sub-images.

C. Comparison Studies

1) *Comparison with Implementation Based on Run-Time Partial Reconfiguration on FPGA:* For comparison purposes, we have implemented the PCNN-based face recognition system consisting of 16 PEs in three configurations (corresponding to three modes) using the run-time partial reconfiguration feature of FPGA device. The memory units, the minimum finding logic, and the external control logic have been implemented as static modules while the systolic array designs without memory units have been implemented as reconfigurable modules. The modules have been separately synthesized for Virtex-4 XC4VFX12 FPGA in Xilinx ISE. The netlists were then imported into PlanAhead Lite 10.1.1 for implementing the partial reconfiguration. The initial configuration for the region was set to the reconfigurable module of Mode-1. The partial reconfiguration was triggered by the change in the mode-control signal during run-time. The partial bitstreams of the reconfigurable modules were stored in the external DDR SDRAM of ML403 platform.

During training phase, the reconfiguration is performed $2IP$ times since Mode-1 and Mode-2 operations are repeated IP times. Since all the PEs need to be reconfigured at the same time, the overlapping between Mode-1 and Mode-2 as seen in Fig. 9(a) should be avoided. This, in turn, increases the training time. The training time (T_T^r) in the case of single-PCNN is estimated to be

$$\begin{aligned} T_T^r &= nIP(3(N+M-1)t_c + t_{r1} + t_{r2}) \\ &\quad + (nPN + M - 1)t_c + t_{r1} \end{aligned} \tag{8}$$

where t_c is the clock period, and t_{r1}, t_{r2}, and t_{r3} are the reconfiguration times for the bitstreams of Mode-1,

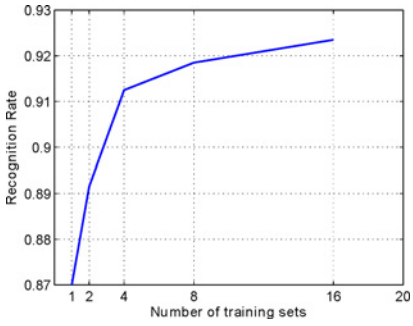

Fig. 13. Recognition performance for the division of training set.

TABLE VI
COMPARISON WITH EXISTING ARCHITECTURES

Description		Architecture in [32]	Architecture in [31]	Proposed
Algorithm	PCA	Weighted modular	Modular	Modular
	Neural network based	No	No	Yes
Architecture	Type	Direct implementation of computations	Direct implementation of computations in multi-lane	Area-time efficient self-configurable systolic architecture
	Eigenface computation	Software	Software	Hardware
	Computation of projections	Hardware	Hardware	Hardware
	Classifier	Hardware	Hardware	Hardware
FPGA synthesis	Device	Altera's EP20K200EFC484	Altera's EP20K600CB652C7	Altera's EP20K600CB652C7
	Training set size	15	120	128 1024
	Logic elements used	7760	19 572	5009 5066
	Maximum clock	33 MHz	91 MHz	75 MHz 75 MHz
	Recognition time	38 ms	1.4 ms	85 μs 0.2 ms
FPGA Implementation	Platform	Altera Stratix II	Nil	Xilinx ML 403
	Handling large training set	–	–	By run-time configuration of BRAMs

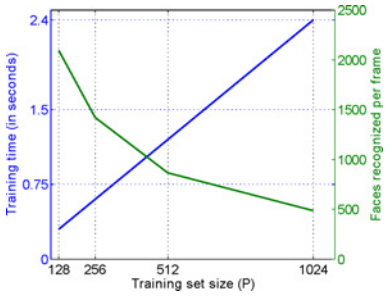

Fig. 14. Performance of FPGA-based recognition system.

Mode-2, and Mode-3, respectively. During recognition phase, the reconfiguration is performed twice, once for each mode of operation. Since there should not be any overlapping between the two modes (Mode-1 and Mode-3), the recognition time (T_R^r) is estimated to be

$$T_R^r = n((N + M - 1)t_c + t_{r1} + (P + M - 1)t_c + t_{r3}). \quad (9)$$

From the implementation results, t_{r1}, t_{r2}, t_{r3}, and t_c were found to be 0.045 s, 0.052 s, 0.049 s, and 7.4 ns, respectively. For $I = 100$, $N = 128$, M=16, $P = 888$ and $n = 8$ as taken in Section VI-A1, T_T^r is more than 10 h and T_R^r is estimated to be 0.75 s. Note that the run-time reconfiguration scheme performs recognition of four faces in three seconds while the proposed architecture performs recognition of 16 518 faces per second using the same FPGA device.

2) *Comparison with Existing Architectures:* A comparison of proposed architecture with the two existing eigenface-based architectures [31], [32] is given in Table VI. All the three face recognition algorithms are based on modular PCA. However, the architectures in [31] and [32] implement only the recognition phase, which involves the computation of projection onto eigenfaces and the classifier. The eigenfaces are precomputed in software and stored in the memory of architecture. The proposed systolic architecture, on the other hand, implements both training and recognition phases. It is neural network-based and therefore it is area-efficient. Results of implementation of the proposed architecture design (for training set sizes 128 and 1024) in Altera's EP20K600CB652C7 device demonstrate the area-efficiency of the solution. The logic elements consumed are much less when compared to the other two architecture implementations even though the proposed architecture implements both training and recognition phases. This is due to simple reusable logic components in the PEs. The architecture can also be operated at a high speed. The recognition time for a new face is estimated to be 85 μs for the similar settings as in [31] [$N = 16 \times 16$, $M = 20$, $P = 128$, and $n = 16$ in C_R^s given by (7)]. The recognition time is much less when compared to others due to high throughput of the systolic architecture.

VII. CONCLUSION

A principal component neural network based face recognition system and its systolic design have been proposed. The proposed system computes eigenfaces and recognizes a new face using the same network. The performance evaluation of the system on Yale and FRGC face databases shows that a recognition rate of more than 85% has been achieved by suitable choice of parameters such as number of neurons and sub-images. The PCNN with linear neurons using the generalized Hebbian learning algorithm facilitates reduced-complexity hardware realization of the face recognition task.

An easily scalable and self-configurable systolic architecture has been designed for efficient hardware realization of the proposed face recognition system. Unlike the existing architectures which involve both software and hardware components, the proposed architecture has been fully implemented in hardware without any software control. Besides, the proposed architecture is different from existing architectures in terms of its ability to update eigenfaces whenever faces are added to or removed from the face database. The self-configurable PEs of the architecture involve minimum resources and have maximum hardware utilization by optimized scheduling of the computations pertaining to different modes of operation.

The proposed systolic design has been implemented and tested on a Xilinx FPGA-based evaluation platform. A systolic array of 16 PEs has been implemented in a Xilinx FPGA

device. From the implementation results we find that the proposed system can recognize more than 400 faces in an image frame with the device storing the features of 1024 enrolled faces of the database. The proposed face recognition system would therefore be quite suitable for real-time surveillance.

REFERENCES

[1] A. J. Goldstein, L. D. Harmon, and A. B. Lesk, "Identification of human faces," *Proc. IEEE*, vol. 59, no. 5, pp. 748–760, May 1971.

[2] A. K. Jain, A. Ross, and S. Prabhakar, "An introduction to biometric recognition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 1, pp. 4–20, Jan. 2004.

[3] Y. Pang, Y. Yuan, and X. Li, "Gabor-based region covariance matrices for face recognition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 7, pp. 989–993, Jul. 2008.

[4] D. C. L. Ngo, A. B. J. Teoh, and A. Goh, "Biometric hash: High-confidence face recognition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 6, pp. 771–775, Jun. 2006.

[5] Y. Fu, Z. Li, J. Yuan, Y. Wu, and T. S. Huang, "Locality versus globality: Query-driven localized linear models for facial image computing," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 12, pp. 1741–1752, Dec. 2008.

[6] M. Liu, S. Yan, Y. Fu, and T. S. Huang, "Flexible X-Y patches for face recognition," in *Proc. IEEE Int. Conf. Audio Speech Signal Process.*, Mar. 2008, pp. 2113–2116.

[7] S. Lucey and T. Chen, "Learning patch dependencies for improved pose mismatched face verification," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2006, pp. 909–915.

[8] K. I. Diamantaras and S. Y. Kung, *Principal Component Neural Networks: Theory and Applications*. New York: Wiley, 1996.

[9] T. D. Sanger, "Optimal unsupervised learning in a single layer feed-forward neural network," *Neural Netw.*, vol. 2, pp. 459–473, 1989.

[10] *Yale Database* [Online]. Available: http://cvc.yale.edu/projects/yalefaces/yalefaces.html

[11] *FRGC Website* [Online]. Available: http://face.nist.gov/frgc

[12] S. Y. Kung, *VLSI Array Processors*. Englewood Cliffs, NJ: Prentice-Hall, 1988.

[13] M. A. Turk and A. P. Pentland, "Eigenfaces for recognition," *J. Cognitive Neurosci.*, vol. 3, no. 1, pp. 71–86, 1991.

[14] A. Lemieus and M. Parizeau, "Experiments on eigenfaces robustness," in *Proc. IEEE Int. Conf. Pattern Recognit.*, vol. 1. Aug. 2002, pp. 421–424.

[15] K. Etemad and R. Chellappa, "Discriminant analysis for recognition of human face images," *J. Opt. Soc. Am.*, vol. 14, no. 8, pp. 1724–1733, Aug. 1997.

[16] X. He, S. Yan, Y. Hu, P. Niyogi, and H. J. Zhang, "Face recognition using Laplacian faces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 3, pp. 328–340, Mar. 2005.

[17] M. Li and B. Yuan, "A novel statistical linear discriminant analysis for image matrix: Two-dimensional fisherfaces," in *Proc. 7th Int. Conf. Signal Process.*, 2004, pp. 1419–1422.

[18] M. Li, B. Yuan, and X. Tang, "Gabor feature based classification using 2-D linear discriminant analysis for face recognition," in *Proc. Int. Conf. Audio Video Based Biometric Person Authentication*, LNCS 3546. 2005, pp. 929–936.

[19] B. Moghaddam, T. Jebara, and A. Pentland, "Bayesian face recognition," *Pattern Recognit.*, vol. 33, no. 11, pp. 1771–1782, Nov. 2000.

[20] A. M. Martinez and A. C. Kak, "PCA versus LDA," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 2, pp. 228–233, Feb. 2001.

[21] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces versus Fisherfaces: Recognition using class specific liner projection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 711–720, Jul. 1997.

[22] M. Sharkas and M. A. Elenien, "Eigenfaces versus Fisherface versus ICA for face recognition: A comparative study," in *Proc. IEEE Int. Conf. Signal Process.*, Oct. 2008, pp. 914–919.

[23] J. Liu and S. Chen, "Discriminant common vectors versus neighborhood components analysis and Laplacianfaces: A comparative study in small sample size problem," *Image Vision Comput.*, vol. 24, pp. 249–262, Mar. 2006.

[24] W. Khan, P. Delmas, G. Gimel'farb, and J. Morris, "Comparing subspace methods for face recognition," in *Proc. Int. Conf. Image Vision Comput.*, 2008.

[25] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE Trans. Neural Netw.*, vol. 8, no. 1, pp. 98–111, Jan. 1997.

[26] M. J. Er, S. Wu, J. Lu, and H. L. Toh, "Face recognition with radial basis function (RBF) neural networks," *IEEE Trans. Neural Netw.*, vol. 13, no. 3, pp. 697–710, May 2002.

[27] J. Lu, X. Yuan, and T. Yahagi, "A method of face recognition based on fuzzy c-means clustering and associated sub-NNs," *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 150–160, Jan. 2007.

[28] P. Melin, C. Felix, and O. Castillo, "Face recognition using modular neural networks and the fuzzy Sugeno integral for response integration," *Int. J. Intell. Syst.*, vol. 20, no. 2, pp. 275–291, Feb. 2005.

[29] Z. Q. Zhao, D. S. Huang, and B. Y. Sun, "Human face recognition based on multi-features using neural networks committee," *Pattern Recognit. Lett.*, vol. 25, no. 12, pp. 1351–1358, Sep. 2004.

[30] J. M. Gilbert and W. Yang, "A real-time face recognition system using custom VLSI hardware," in *Proc. IEEE Workshop Comput. Architectures Mach. Perception*, May 2003, pp. 58–66.

[31] R. Gottumukkal and V. K. Asari, "Multi-lane architecture for eigenface based real-time face recognition," *J. Microprocessors Microsyst.*, vol. 30, no. 4, pp. 216–224, 2006.

[32] A. P. Kumar, V. Kamakoti, and S. Das, "System-on-programmable-chip implementation for on-line face recognition," *Pattern Recognit. Lett.*, vol. 28, no. 3, pp. 342–349, Feb. 2007.

[33] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Englewood Cliffs, NJ: Prentice-Hall, 1999.

[34] P. J. Phillips, P. J. Flynn, T. Scruggs, K. W. Bowyer, J. Chang, K. Hoffman, J. Marques, J. Min, and W. Worek, "Overview of the face recognition grand challenge," in *Proc. IEEE Conf. CVPR*, vol. 1. Jun. 2005, pp. 947–954.

[35] R. Gottumukkal and V. K. Asari, "An improved face recognition technique based on modular PCA approach," *Pattern Recognit. Lett.*, vol. 25, no. 4, pp. 429–436, Mar. 2004.

N. Sudha (M'99–SM'07) received the B.E. degree from Madurai Kamaraj University, Madurai, India, in 1992, the M.S. degree from the Indian Institute of Technology (IIT) Madras, Chennai, India, in 1997, and the Ph.D. degree from IIT Guwahati, Guwahati, India, in 2001.

She has been an Assistant Professor with the School of Computer Engineering, Nanyang Technological University, Singapore, since 2006. Her current research interests include image processing, embedded systems, and neural networks.

A. R. Mohan (S'09) received the B.E. degree in electronics and communication engineering from the College of Engineering Guindy, Anna University, Chennai, India, in 2005. Currently, he is pursuing the Ph.D. degree with the School of Computer Engineering, Nanyang Technological University, Singapore.

His current research interests include computer vision, biometrics, and very large scale integration architectures.

Pramod K. Meher (SM'03) received the B.S. and M.S. degrees in physics and the Ph.D. degree from Sambalpur University, Sambalpur, India, in 1976, 1978, and 1996, respectively.

Currently, he is a Senior Scientist with the Department of Embedded Systems, Institute for Infocomm Research, Singapore. He holds an adjunct position with the School of Computer Engineering, Nanyang Technical University, Singapore. His current research interests include design of dedicated and reconfigurable architectures for computation-intensive algorithms in signal and image processing, communication, and bioinformatics.