REFRESH: REDEFINE for Face Recognition using SURE Homogeneous Cores

Gopinath Mahale, Hamsika Mahale, S.K.Nandy, *Senior Member, IEEE,* Ranjani Narayan

Abstract—In this paper we present design and analysis of a scalable real-time Face Recognition (FR) module to perform 450 recognitions per second. We introduce an algorithm for FR, which is a combination of Weighted Modular Principle Component Analysis and Radial Basis Function Neural Networks. This algorithm offers better recognition accuracy in various practical conditions than algorithms used in existing architectures for real-time FR. To meet real-time requirements, a Scalable Parallel Pipelined Architecture (SPPA) is developed by realizing the above FR algorithm as independent parallel streams and sub-streams of computations. SPPA is capable of supporting large databases maintained in external (DDR) memory. By casting the computations in a stream into hardware, we present the design of a Scalable Unit for Region Evaluation (SURE) core. Using SURE cores as computer elements in a massively parallel CGRA, like REDFINE, we provide a FR system on REDEFINE called REFRESH. We report FPGA and ASIC synthesis results for SPPA and REFRESH. Through analysis using these results, we show that excellent scalability and added programmability in REFRESH makes it a flexible and favorable solution for real-time FR.

Index Terms—Face and gesture recognition, Real-time systems, Parallel Architectures, Reconfigurable hardware, Multi-core/single-chip multiprocessors

1 Introduction

THE field of biometrics has changed the manner in which identities are perceived in the real world. Ranging from accurate finger print recognition to recent iris recognition, individuals are recognized beyond human capacity. Inclusion of these automated biometric methods in security systems has been popular these days to avoid manual human handling errors. Although various biometric methods are in use, reliability of these systems in critical applications is a matter of concern. Face Recognition (FR) is one such biometric method to identify individuals from features present in their face images. This non-contact based system is preferred over other biometric methods due to its ease of use and ability to recognize without the knowledge of the subject. FR system has numerous applications which include human-computer interaction, authentication, surveillance etc. FR system is also used in law enforcement applications such as identification of crime suspects in crowds. The factors that influence accuracy of FR in the real world are variations in illumination, pose angle, facial expressions, occlusions, and variations due to ageing of subjects. For critical applications of FR, such as identification of crime suspects in crowd, real-time response is very much necessary. The latency in recognition is affected by complexity of algorithm being used for recognition. Although there are numerous algorithms [1] [2] [3] proposed for FR, it has been hard to achieve high frame-rate real-time FR for efficient, but compute intensive, algorithms. As available software solutions for FR have significantly large latency in recognizing individuals, they are not suitable for real-time applications. Existing hardware solutions in literature use simple classifiers to reduce the complexity for real-time performance, resulting in poor recognition accuracy. In addition, due to limited expensive on-chip memory resources, these architectures limit the database size to a small number. Therefore there is a need for a scalable real-time architecture for FR module that ensures good recognition accuracy, large number of recognitions per second and targets very large databases.

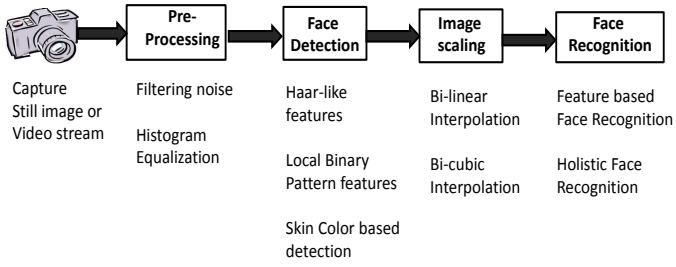


Fig. 1. A general face recognition system

A block schematic of various stages of a real-time FR system is given in Fig. 1. Input image or a frame of video stream, which may consist of more than a single face, is captured by a camera. An optional pre-processing stage performs filtering of noise or nullification of illumination changes. The locations of faces in the input image, which are regions of interest, are detected by a face detection module. These images of detected faces in different sizes are resized by an image scaling module to the size specified by the face recognition module following it. The final block of FR system, a FR module, extracts features in the input image and matches it against images available in the database.

In this paper, we design a scalable real-time FR module with good recognition accuracy that targets databases of large sizes. The first step towards design of scalable FR module is the selection of FR algorithm. We explore algorithms with good recognition

- *G. Mahale, H. Mahale and S.K.Nandy are with Computer Aided Design Laboratory, Indian Institute of Science, Bangalore, India E-mail: {mahalegv,hamsika,nandy}@cadl.iisc.ernet.in*
- *R. Narayan is with Morphing Machines Pvt. Ltd., Bangalore, India E-mail: ranjani@morphing.in*

Manuscript received April 19, 2005; revised September 17, 2014.

accuracy and modularity suitable for real-time applications. We introduce a combination of Weighted Modular Principle Component Analysis (WMPCA) and Radial Basis Function Neural Network (RBFNN) [4], which is found to show better recognition accuracy than algorithms used in existing hardware solutions in various practical conditions. According to the industry standards, a frame rate of 15 per second is sufficient to capture public movement in crowded places [5]. We target to recognize at most 30 faces in each of these frames, which amounts to 450 recognitions to be performed per second. To achieve real-time performance at this frame rate, we exploit modularity in WMPCA to come up with a Scalable Parallel Pipelined Architecture (SPPA) [4] for real-time FR. The parallelism in our FR algorithm is exploited at the level of image regions using streams of computation. Parallel sub-streams under each stream are realized to perform computations in corresponding regions. The data is stored off-chip for scalability with respect to database sizes. In addition, a novel data layout is used to store data on off-chip memory to achieve good memory throughput. The SPPA is emulated on Virtex-6 LX550T FPGA for functional verification. The emulated system operating at 107 MHz is able to perform 450 recognitions per second on databases containing at most 450 classes. For scalability and flexibility in usage, we extend this design to a reconfigurable multi-core environment such as REDEFINE [6]. Reconfigurability in REDEFINE allows composition of custom data paths at run-time, which serve as accelerators for certain critical application specific functionalities. We dwell more on this in section 5. We exploit parallelism in the algorithm to distribute computations over the nodes of REDEFINE. To achieve real-time FR, each processing node in REDEFINE is enriched with embedded hardware accelerator core capable of performing macro operations in hardware. Such an accelerator core performs computations that are analogous to the computations performed in a single stream of SPPA. We refer to such an accelerator core as Scalable Unit for Region Evaluation (SURE) core. Such multiple SUREs connected by the NoC constitutes our scalable FR module which we term REFRESH, REDEFINE for Face Recognition using SURE Homogeneous cores. REFRESH with 16 SUREs, emulated on Virtex-6 FPGA operating at 100MHz, is capable of performing 450 real-time recognitions per second on a face image database consisting 417 classes. The number of recognitions per second, number of classes and image dimensions supported by REFRESH scale with number of SUREs. FPGA emulation of REFRESH offers performance that is almost equal to that of SPPA on FPGA. The negligible degradation in performance of REFRESH compared to SPPA, is a price easily borne for meeting the larger objectives of programmability and scalability.

The rest of the paper is organized as follows. In section 2, we explore algorithms in literature suitable for real-time implementation. In addition, we discuss in brief the existing hardware and software solutions for real-time FR from literature. In section 3, we summarize the FR algorithm from [4]. In addition, we analyse its recognition accuracy on images with occlusion and report its performance with different clustering methods. In section 4, we reproduce from [4] a scalable modular parallel pipelined architecture for the FR algorithm along with FPGA and ASIC synthesis results and comparison against existing real-time architectures for FR. In section 5, we describe our extended work in SURE based acceleration on REDEFINE with emulation on FPGA and ASIC synthesis results. We conclude the paper in section 6.

2 Algorithms for FR and Existing FR systems

Significant research is being pursued towards design of efficient algorithms for face recognition. However, it has been hard to find an algorithm that targets all the factors affecting its recognition accuracy. Considering trade-off between recognition accuracy and computation complexity, we look for algorithms with good recognition accuracy and scope for hardware acceleration of parallel computations. The choice of algorithm is also done by considering the ease with which the FR module can be trained on-line.

In this exposition we restrict to two dimensional FR algorithms as three dimensional FR deals with complicated camera set-up and compute intensive algorithms. In two dimensional FR, feature based FR algorithms [3] detect individual features in the input image of face and perform recognition. Recognition performance of these algorithms depends on accuracy of feature detection. In addition, the compute intensiveness of these algorithms make them not a suitable option for real-time applications. On the other hand, holistic FR algorithms [3] use the whole input image of face for recognition of individual. As holistic FR algorithms are simple and very well suited for real-time implementations, we consider them in our algorithm exploration for the designed FR module. To recognize faces, holistic FR algorithms include two steps.

The first step is extraction of features in the input image. The goal of feature extraction is to use only significant information in the input image, thereby reducing the dimension of input vector. Principle Component Analysis (PCA) [7], Linear Discriminant Analysis (LDA) [8], wavelet decomposition [9] are some of the popular feature extraction methods. PCA is a popular technique used in extraction of discriminating features in the input data. It has been experimentally shown that, with respect to recognition accuracy, LDA is outperformed [8] by PCA in practical conditions. Some major modifications to PCA, such as Modular PCA (MPCA) [10], Weighted Modular PCA (WMPCA) [11], have shown better feature extraction capabilities. PCA along with these modifications are discussed in detail in [4].

The second step in holistic FR algorithm is to recognize the input image using the feature vector made of features extracted from the input image. This process is termed classification. An example for a simple classifier is Nearest Neighbour Classifier (NNC) with rectilinear or Euclidean distance as distance measures. Being the simplest classifier, NNC has been used in many real-time FR systems due to its lower complexity. However, due to its greedy nature, NNC often results in misclassification. There exist sophisticated classifiers such as RBFNN, Multi-Layer Perceptrons (MLP) and Support Vector machines (SVM), which have shown better classification abilities compared to that of NNC. We use RBFNN in the algorithm for real-time FR due to the following reasons.

- RBFNNs are well suited for pattern recognition problems and it is stated that due to locally tuned processing nodes, it provides scope for faster learning [12]
- RBFNNs with just one hidden layer can achieve universal approximation [13]
- RBFNNs are shown to exhibit shift, scale and pose invariance after training [14]
- RBFNN shows better generalization performance and are computationally faster than SVM for large training data sets [15]

PCA with RBFNN has shown better recognition accuracy than wavelet decomposition with RBFNN [16]. In addition, availability of simpler incremental training algorithms [17] for RBFNN has encouraged us in using it for our real-time FR module [18]. The computations involved in RBFNN are described in [4]. Although SVM with Gaussian kernel resembles RBFNNs with Gaussian activation function [15], they differ in selection of mean and variances of clusters and training of weights. RBFNN has shown better classification abilities compared to SVM in some pattern recognition problems [19]. Incremental training of SVM does not show much scope for on-line real-time hardware realizations. Enormous amount of computations involved in training and classification in deep neural networks [20] and convolutional neural networks [21] have restricted us from considering it for our FR module. Similarly, dictionary based FR algorithm using sparse representation [22] involves large amount of computations which make it not a suitable candidate for real-time systems. There are probabilistic approaches for FR such as using Hidden Markov Models [23] which have not shown considerable improvements over other techniques.

There has been substantial work towards design of real-time FR. A real-time face tracking and FR using RBFNN was realized by Yang et al. [24] on FPGA, ZISC processor and DSP processor to get 14, 25 and 4.5 recognitions per second respectively. A near real-time Embedded FR proposed by Zuo et al. [25] with feature based FR algorithms was able to work at 4 frames per second on a Pentium 4 processor. Rajkiran Gottumukkal et al. [26] have proposed a MPCA based FRS with a fixed number of 20 Principle Components (PCs) and input image down sampled to a very small size. A WMPCA and NNC based FR module as a system on programmable chip with 26 recognitions per second was proposed by Pavan Kumar et al. [27]. Visakhasart et al. [28] have proposed a multi-pipeline PCA and NNC based FR module for a very small face database. Janarbek et al [29] proposed a PCA and NNC based FR system which could recognize 45 faces per second with input images down sampled to 20×20. A MPCA and NNC based self configurable systolic architecture was proposed by Sudha et al. [30]. A very large number of recognitions per second is reported for this architecture. However, for large number of training images in database, throughput is affected by transfer of data from off-chip memory to on-chip Block RAMs. In addition, classification by NNC and use of low resolution images for better throughput are downsides of the architecture. We conclude from the literature that the existing FR systems either use simple algorithms to meet real-time requirements or use small databases due to limited on-chip memory resources. In addition, the architectures are not scalable with growing image dimensions, number of classes, number of recognitions per second etc. These growing requirements necessitate design of a scalable hardware architecture for FR. In the following sections we focus on a modular algorithm and a hardware architecture to accelerate it for real-time FR that address accuracy and scalability issues in the existing architectures.

3 A Hybrid Algorithm: Combined WMPCA and RBFNN algorithm

In [4] we build on the algorithm for FR which is a combination of WMPCA and RBFNN. Recognition performance of our algorithm on AT&T Lab (ORL) [31], Sheffield database (UMIST) [32] and Extended Yale database B [33] is discussed. In addition, a real-time parallel pipe-lined architecture for the algorithm and emulation results are reported. It can be concluded about the algorithm from [4] that

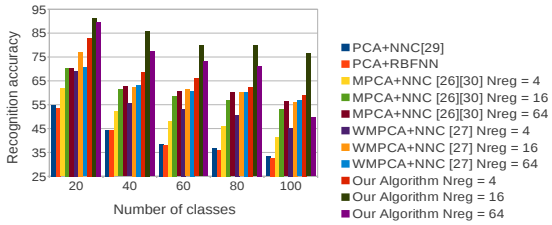


Fig. 2. Recognition accuracy of algorithms on different databases as observed on MATLAB

- The algorithm shows good recognition accuracy on images with considerable variations in pose, expression and illumination conditions
- Increase in the number of regions has positive impact on recognition accuracy
- Compared to algorithms used in existing architectures for real-time FR, we see improved recognition accuracy by our algorithm even with large number of classes.
- Modularity in the algorithm makes it suitable for acceleration by parallel processing in real-time applications

We assume that input to our algorithm is output from a face detector module that detects and normalizes faces from a video frame or a still image. We evaluate our algorithm on AR face database [34] which contains images with illumination variations and images with occlusion by sun glass and muffler scarf. Using this database, we analyse recognition accuracy of our algorithm in law enforcement applications such as criminal identification in crowded places.

From each class of AR database, first seven regular images were used for training and rest 19 images with occlusion, change in illumination and expression were used as test images to mimic the scenario of criminal identification in public. For PCA with RBFNN, whole images were used for feature extraction and resultant feature vectors were classified by RBFNN. μ and σ [4] for the hidden nodes were chosen as mean of samples from each class and average distance of samples from the mean respectively. Thus, number of hidden nodes is equal to the number of classes. For NNC, Euclidean distance measure was used for similarity measure. We fix the number of Principle Components (PC) to 32 [4]. In Fig. 2 we plot recognition accuracy of algorithms used in existing architectures against the number of classes for AR face database. For AR database, the algorithm works much better on images with variations in illumination, expression and occlusions than rest of the algorithms. Increasing the number of regions results in smaller image regions which gives rise to vectors of smaller dimensions. This leads to smaller domain of classification resulting in better classification. However, in the case of AR database, due to occlusion of faces, smaller regions are not able to capture discriminating features from the input image. Due to this, we see a decrease in recognition accuracy for very large number of regions.

The selection of mean, μ, and standard deviation, σ, of clusters in the feature space [4] can be done in any of the following methods.

- By supervised clustering with number of hidden nodes equal to number of classes
- By unsupervised clustering of feature space
- By intra-class clustering

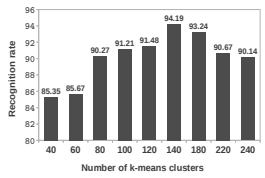
Supervised clustering by making number of hidden nodes equal to the number of classes is simplest among the listed methods, as there are no iterative computations involved. μ for a hidden node is equal to the mean of samples belonging to the corresponding class. σ for a hidden node is taken as the average distance of samples from the corresponding mean. The results shown in Fig. 2 and [4] follow this method of clustering the data. The disadvantage of this method is that it may cause cluster overlapping.

Unsupervised clustering of feature space is accomplished by applying algorithms such as k-means clustering [24] on the entire feature space. Using this method, we can create arbitrary number of clusters, where samples from two different classes may also fall under a single cluster. Basically k-means clustering is an iterative algorithm whose outcome depends on the initialization conditions. Therefore, to get an optimal clustering, the algorithm needs to be run a number of times to select the best clustering. Finally a supervised training is performed on the output layer of RBFNN to tune the weights accordingly. Fig. 3(a) shows plot of recognition accuracy against the number of k-means clusters for Extended Yale database B. We observe that, recognition accuracy increases with increase in the number of k-means clusters. This is because of reduced confusion in classification due to better separated clusters. However, recognition accuracy falls beyond a certain point due to overfitting of data.

If each class is assigned a single cluster, they may overlap and samples may not be classified accurately. To address this problem, samples under each class are clustered to get more than one clusters. We term this process intra-class clustering and we call such clusters under each class as sub-clusters. Number of methods have been proposed in literature for such intra-class clustering [35] [36] [24]. In our experiment, we perform k-means clustering on each class whose clusters may otherwise overlap with a single cluster per class. Fig. 3(b) shows two well separated classes due to sub-clustering. Finally synaptic weights [4] of output layer are trained accordingly. For demonstration, we plot recognition accuracy of our algorithm against number of sub-clusters per class for Extended Yale Database B in Fig. 3(c). Here too we observe improvement in recognition accuracy with number of sub-clusters per class, which is superior to that observed for unsupervised clustering over the entire feature space. Here too we see decline in recognition accuracy due to overfitting of data beyond a limit.

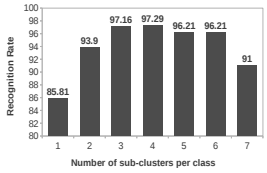
Recognition accuracies reported in Fig. 2 and in [4] are observed in experiments where the output of FR module is the index of class with maximum similarity with the test image. However, in practical conditions, FR module needs to reject the subject, when it is not available in the database. In the case of NNC, it can be implemented using a distance threshold [30]. For RBFNN, such images are rejected by training the system with negative samples and hence creating a set of negative clusters. Assigning more than one cluster for negative samples helps in achieving better classification. Fig. 3(d) shows recognition accuracy of our algorithm on Extended Yale Database B with negative class. Out of 37 classes with 40 samples each, alternate images from first 10 classes are used for training and 40 samples from the next 20 classes are used as negative samples. Rest of the images in the database are used for testing. Here we consider both positive and negative samples for testing to verify performance of our algorithm for recognition of crime suspects in crowd. We perform k-means clustering only on negative class to get more than one cluster for it. We see that increasing the number of sub-cluster under negative class improves recognition accuracy. The recognition accuracy is affected beyond a point due to overfitting of data. Recognition accuracy of the FR module depends on the number of negative samples used and the type of negative samples used. Here performance of FR module also depends on the number of clusters assigned for negative samples. Therefore, for negative class classification we do not compare our algorithm with rest of the algorithms mentioned in Fig. 2.

We conclude that our algorithm works well on images of faces with variation in pose, illumination, expression and also on occluded images. Recognition accuracy of algorithm is improved by assigning more than one cluster for each class. In addition, the algorithm has shown good performance in rejecting images not present in the database. To support real-time FR using our algorithm there is a need for a scalable FR module which can accommodate images of large dimensions, large number of clusters and sub-clusters. We find the required parallelism for acceleration in the modular computations of WMPCA, where the computations in each region can be performed in parallel [4]. We further exploit parallelism in computations under each region to come up with a Scalable Parallel Pipelined Architecture (SPPA) which can be used with any of the mentioned clustering techniques. In the next section we describe SPPA for our algorithm in detail and report details of realization on hardware.

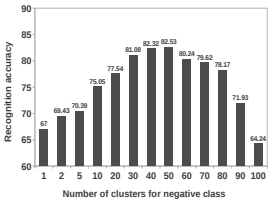


(a) Recognition accuracy for Extended Yale Database B with different number of clusters formed by k-means clustering

(b) Intra-class clustering



(c) Recognition accuracy for Extended Yale Database B with different number of sub-clusters under each class formed by k-means clustering



(d) Recognition accuracy for Extended Yale Database B with different number of sub-clusters for negative class

Fig. 3. Recognition accuracy by clustering and sub-clustering

4 Scalable Parallel Pipelined Architecture(SPPA) for Real-time FR

SPPA for FR is designed to meet the need for recognition of large number of faces per second and support for very large databases.

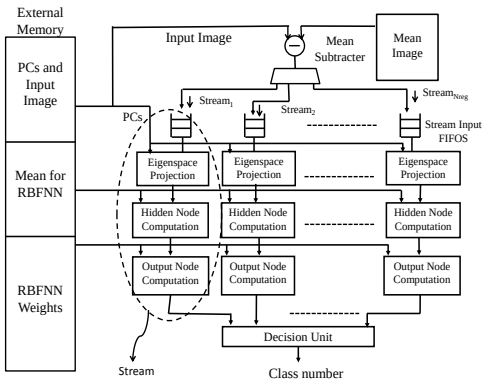
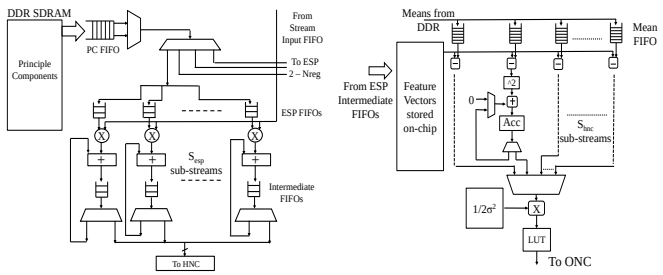


Fig. 4. Architecture of SPPA for real-time FR

Acceleration is brought about by exploiting parallel operations in the algorithm explained in [4]. Modularity in the algorithm is effectively utilized to come up with parallel and independent streams and sub-streams of computation. We realize independent computations in each region as streams. Parallel operations in each stream are implemented as sub-streams of computations.

We reproduce the architecture of SPPA from [4] in detail. Fig. 4 shows overall view of the architecture. We store the mean image, RBFNN variances and look-up table for exponentiation in on-chip memory. The Input image, PCs, RBFNN means and RBFNN synaptic weights are stored in off-chip RAM. A detailed explanation on data layout for off-chip RAM is given in section 4.5. There are N_{reg} number of streams each performing computations in respective region. Initially for mean subtraction, input image pixels from off-chip RAM are subtracted from corresponding mean image pixels yielding mean subtracted pixels, that are forwarded to stream of corresponding region. A single subtracter performs mean subtraction for all the regions in sequence. Computations in each stream are divided into three stages namely Eigenspace Projection (ESP), Hidden Node Computation (HNC) and Output Node Computation (ONC). In ESP, the mean subtracted pixels are projected on eigenspace computed during training. In HNC, the RBFNN hidden layer outputs are computed which involves computing Gaussian outputs. In ONC, RBFNN synaptic weights are multiplied with hidden node outputs to get corresponding RBFNN outputs. Decision Unit (DU) combines outputs of all the streams to perform final classification.



(a) Architecture of ESP unit (b) Architecture of HNC unit

Fig. 5. Architectures of Eigen Space Projection and Hidden Node Computation of SPPA

4.1 Eigenspace projection

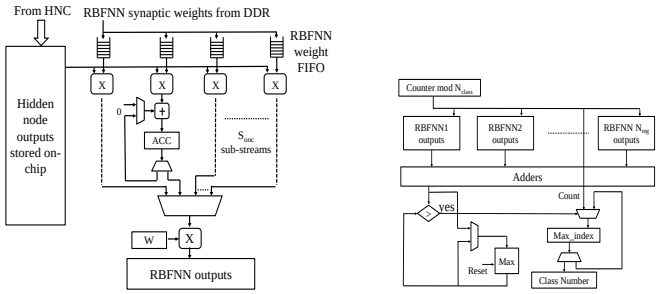
Architecture of ESP is shown in Fig. 5(a). Operation to be performed in ESP is basically multiplication of mean-subtracted image vector with eigenspace matrix consisting of PCs to get feature vector of reduced dimension [4]. The PC's stored in external memory are fetched in bursts and distributed among N_{reg} number of streams. These PCs are in turn distributed among FIFOs, termed ESP FIFOs, of sub-streams. Each stream in ESP consists of S_{esp} number of sub-streams to perform independent computations in parallel. Each such sub-stream has a multiplier, an adder and a FIFO termed Intermediate FIFO to store the intermediate results. Eigenspace matrix has N_{pc} number of PCs arranged as columns. If this vector matrix multiplication is performed in regular fashion, i.e., multiplication of mean subtracted pixel with each column of eigenspace matrix, then it would need reading of mean subtracted pixels multiple times. Otherwise, we would need a storage space of $M \times N$ locations to store the mean subtracted image pixels. To address this issue and to achieve maximal data locality in our architecture, we fetch the elements of eigenspace matrix in row-wise manner. Algorithm 1 in Appendix shows the manner in which the multiplications are performed. These computations result in 32 intermediate results due to 32 number of PC's used in our FR algorithm. These are stored in on-chip intermediate FIFOs and taken out once the multiplication of mean subtracted pixels in a region with all the PC's are performed. As these computations are divided among sub-streams under each stream, the depth of each intermediate FIFO is equal to $\left\lceil \frac{N_{pc}}{S_{esp}} \right\rceil$. Outputs of ESP are stored in a local memory and this data is further consumed by HNC.

4.2 Hidden Node Computation

HNC computes the hidden node outputs of RBFNN which includes sum of squared differences between the feature vector and RBFNN means. This result is divided by $2 \times \sigma^2$, where σ is standard deviation of RBFNN cluster. Exponentiation of this result will give the output of corresponding hidden node [4]. Architecture of HNC is shown in Fig. 5(b). There are S_{hnc} sub-streams of computations each consisting of a subtracter, squaring unit, an adder and an accumulator. The RBFNN means stored in off-chip memory are read in bursts and are distributed among streams and S_{hnc} number of sub-streams. The received RBFNN means in the mean FIFO are subtracted from feature vectors computed during ESP and are stored in a local memory. The outputs are squared and accumulated. Algorithm 2 in Appendix describes computations involved in detail. The accumulated outputs are sequentially routed out of sub-streams for division by $2 \times \sigma^2$. This division is implemented as multiplication with $\frac{1}{2 \times \sigma^2}$, which is precomputed and stored in an on-chip memory. Exponentiation is performed on the resultant value using a locally stored 13 bit look-up table. The resulting hidden node outputs are stored in a local memory and are consumed by ONC for multiplication with synaptic weights.

4.3 Output Node Computation

Multiplication of HNC outputs with synaptic weights is performed by S_{onc} number of sub-streams as shown in Fig. 6(a). Each sub-stream consists of a multiplier and an adder. Basically this computation is also a multiplication of hidden node output vector with synaptic weight matrix. The synaptic weights are read from off-chip memory in the form of bursts and are distributed among



(a) Architecture of ONC unit (b) Architecture of DU

Fig. 6. Architectures of Output Node Computation and Decision Unit of SPPA

TABLE 1
Device Utilization of SPPA on Virtex-6 LX550T FPGA ; $RecPS$=450, M=128, N=128, N_{pc}=32, N_{class}=450 and N_{clust}=450. For N_{reg}=16, S_{esp}, S_{hnc}, and S_{onc} are 2, 2 and 8 respectively and that for N_{reg}=4 are 8, 2 and 8 respectively

	SPPA, N_{reg} = 16	**SPPA, N_{reg} = 4**	**PCA and RBFNN**
Number of slice registers	138987 out of 687360 (20%)	74231 out of 687360 (10%)	59652 out of 687360 (8%)
Number of slice LUTs	168245 out of 343680 (48%)	62874 out of 343680 (17%)	39149 out of 343680 (36%)
Number of Block RAMs	265 out of 632 (41%)	239 out of 632 (37%)	228 out of 632 (36%)
Number of DSP48Es	176 out of 864 (20%)	44 out of 864 (5%)	19 out of 864 (2%)

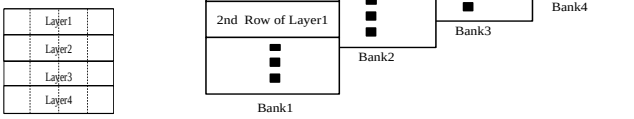
streams and sub-streams. The synaptic weights in RBFNN weight FIFO are multiplied with HNC outputs stored in the on-chip memory as described in Algorithm 3 in Appendix. After multiplication with all the corresponding synaptic weights, the results in the accumulator are taken out in sequence and are multiplied with region weights. These ONC outputs are forwarded to DU for the final classification.

4.4 Decision Unit

The decision unit combines the outputs of N_{reg} streams and performs the final classification. Architecture of decision unit is shown in Fig. 6(b). Output vectors of all RBFNNs are added to get combined output which is used in final classification by finding the maximum. Maximum among the vector elements is found by sequentially comparing the values as shown in Fig. 6(b). The output of DU, Max_index, gives the recognized class.

4.5 Data layout

As an example we have taken input image dimensions equal to 128×128, 32 principle components, 500 classes and single cluster per class. For these specifications, with data representation by 32 bits, we need more than 17 MB of memory to store input image, eigenspace, RBFNN mean and RBFNN synaptic weights. On-chip SRAM storage for memory of this size results in utilization



(a) Input image divided into 16 regions

(b) Input image and PC values stored in four memory banks

Fig. 7. Data layout for ESP in DDR3 SDRAM for N_{reg}=16

of expensive on-chip resources as well as leads to high power dissipation. On the other hand, off-chip Synchronous Dynamic RAMs (SDRAMs) provide large storage space and the memory throughput can be very well utilized with supporting data layout. In the SPPA for FR, we use DDR3 SDRAM as external memory. In DDR3 SDRAM, switching rows in a bank results in more latency than accessing data from different banks. Therefore in order to achieve high memory throughput, we deduce a data layout for elements in the database, which results in minimal intra-bank row changes. In addition, the data layout supports writing of input image to the external memory in bursts for continuous operation of FR module. We store the mean image, RBFNN variances and look-up table for exponentiation on on-chip memory.

The data from external memory need to be distributed among streams and sub-streams of designed FR module. Due to modular processing in WMPCA, computations of different regions happen in parallel. To facilitate distribution of data to corresponding streams and also to achieve minimal intra-bank row changes, data to be distributed in parallel are stored in separate memory banks. We term group of adjacent regions a layer. Fig. 7(a) shows division of input image into 4 layers for N_{reg} equal to 16. For ESP, pixel values of input image and associated PCs from different layers are stored in separate memory banks as shown in Fig. 7(b). In each memory bank shown, a burst of PCs is stored before a row of input image pixels from each layer. This is done to avoid wait states in ESP by making the PCs available in ESP FIFO before mean subtracted pixels reach ESP for multiplication. Following this, PCs from different regions in the corresponding layer are stored in Round-robin arbitration as one burst at a time. This form of storage helps us in maintaining approximately uniform distribution of data across streams and also in reducing amount of data buffering in each stream. The same pattern of storage is followed for rest of the rows of input image and corresponding PCs in each layer. For HNC, the order of data usage is dependent on the value of S_{hnc}, which is described by Algorithm 2 in Appendix. Here too, data to be supplied to each region is written to a bank of memory in the order of access in Round-robin arbitration as one burst at a time. Similarly for ONC, a burst of RBFNN synaptic weights each from all the regions are stored in a bank of memory. Thus, for N_{reg} equal to 16, we use six memory banks of DDR3 SDRAM. First four memory banks are used to store input image and PCs. The next two memory banks are used to store RBFNN means and synaptic weights respectively. By this data layout we are able to achieve high memory throughput for real-time FR module.

TABLE 2
ASIC synthesis results of SPPA on Cadence RTL Compiler; $RecPS$=450, M=128, N=128, N_{pc}=32, N_{class}=450 and N_{clust}=450. For N_{reg}=16

Maximum Operating Frequency	200 MHz
Area	5.6 mm^2
Power	4.4 W

4.6 Synthesis and Performance Analysis of SPPA

Each stream in SPPA is divided into a 3 stage pipeline. The data set accessed by each stage is kept separate by the design so that the stages can operate concurrently. In addition, the division is done in order to make the SPPA scalable with respect to image dimensions, number of classes and number of PCs, number of clusters per class etc. DU is also kept as one of the pipe-line stages to incorporate future enhancements to the decision logic. Number of clock cycles taken by each of these stages is given by,

$$T_{ESP} = n_{esp} \times \left\lceil \frac{N_{pc}}{S_{esp}} \right\rceil \times \left(\frac{MN}{N_{reg}} \right) \quad (1)$$

$$T_{HNC} = n_{hnc} \times N_{pc} \times \left\lceil \frac{N_{clust}}{S_{hnc}} \right\rceil \quad (2)$$

$$T_{ONC} = n_{onc} \times (N_{clust} + 1) \times \left\lceil \frac{N_{class}}{S_{onc}} \right\rceil \quad (3)$$

$$T_{DU} = N_{class} + log_2 N_{reg} \quad (4)$$

Here, n_{esp}, n_{hnc} and n_{onc} are the number of clock cycles required to perform an operation in the corresponding stage. To make it a balanced pipeline and to compute the number of sub-streams required in each stage, these values are equated to the number of clock cycles allotted for each input image for a given recognitions per second, $RecPS$.

$$T_{ESP} = T_{HNC} = T_{ONC} = T_{DU} = \frac{Operating\ frequency}{RecPS} \quad (5)$$

Increasing sub-streams does not improve recognitions per second beyond a certain point due to limited off-chip memory bandwidth. By having a memory with better memory throughput or having different memories for different stages of streams will increase the throughput beyond this limit.

For functional verification, SPPA is simulated and then synthesized on Xilinx ISE targeting M503 module of Pico computing system [37]. M503 has a Xilinx Virtex-6 LX110T FPGA and 2 external DDR3 SDRAMs of 4GB each. For this experiment, we have fixed the image dimension to 128×128, N_{class} to 450 and N_{pc} to 32. Hardware specifications are written in Bluespec System Verilog [38] to generate Verilog specifications. We fix N_{reg} equal to 16 beyond which availability of hardware resources becomes a limiting factor for real-time performance. We use fixed point arithmetic in this architecture which is faster and consumes lesser hardware resources than floating point arithmetic. The number of bits used to represent values are chosen such that we do not lose out on accuracy. Mean image pixels and PCs are represented using 16 bits. 32 bits are used for RBFNN mean, variance and synaptic weights. DSP48E's are used to perform multiplications in each stage. Block RAMs are used for large sized FIFOs and buffers between stages. On Pico computing platform with M503 modules, a C++ program is run on the host machine which writes the input image to predefined locations on DDR3 SDRAM. After recognition, the recognized class is displayed by the host program.

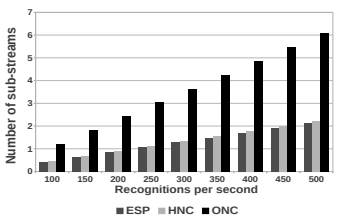


Fig. 8. Recognitions per second of SPPA for different number of sub-streams, N_{class} = N_{clust} = 450, N_{reg}=16, N_{pc} = 32, M = 128, N = 128

Although the number of sub-streams can be increased for higher number of recognitions per second or number of classes, the performance is limited by the memory throughput provided by the external memory. By fixing the number of recognitions to be performed per second to 450, we compute number of classes that our FR module can support with image of dimensions 128×128 and 32 PCs using equation 6.

$$\begin{aligned}(M \times N \times (B_i + N_{pc} \times B_{pc}) + (N_{pc} \times N_{clust} \times B_m \\ +(N_{clust} + 1) \times N_{class} \times B_w) \times N_{reg}) \\ \times \frac{1}{2 \times Data\_width} = \frac{\beta \times Freq_{DDR}}{RecPS}\end{aligned} \quad (6)$$

Here, B_i, B_{pc}, B_m and B_w are the number of bits used to represent input image pixels, PCs, RBFNN means and synaptic weights respectively. DDR3 SDRAM on M503 module has a data width, $Data\_width$, of 64 with an operating frequency of 400 MHz. We use a factor β equal to 0.9 to consider the clock cycles consumed for row changes and periodic refreshes of DDR3 SDRAM. By substituting these values in equation 6, assuming single cluster per class, we get N_{class} equal to 450.

We have realized two configurations of SPPA with N_{reg} equal to 4 and 16 respectively on FPGA. In addition, for comparison, a FR module using PCA and RBFNN is also developed, where acceleration is brought about by a number of sub-streams. Table 1 shows increase in device utilization with increase in N_{reg}. Synthesis results show maximum operating frequency to be 108 MHz. This frequency does not change with change in sub-streams in each stream. Using equation 5 we plot Recognitions per second against number of streams in each stage as shown in Fig. 8. From the plot, we observe that to achieve 450 recognitions per second, we need to have S_{esp}, S_{hnc} and S_{onc} equal to 2, 2, and 8 respectively. To support higher dimension images or more number of PCs, S_{esp} can be increased. Similarly to support more number of clusters or classes S_{hnc} and S_{onc} respectively can be increased. But finally image dimensions, N_{pc}, N_{class} and N_{clust} are limited by external memory throughput for fixed $RecPS$. We also synthesized SPPA at 65 nm technology on Cadence RTL Compiler. Table 2 lists the maximum operating frequency, power and area of the synthesized module

Table 3 compares SPPA with existing architectures for real-time FR. The algorithm used in SPPA is shown to be superior to other algorithms used in the existing architectures. SPPA is not constrained by available on-chip memory resources due to the availability of large sized external memory. For a desired $RecPS$, as seen from Table 3, dimensions of image used in SPPA is much larger than images used in other architectures. This in turn helps in better recognition of subjects due to improved quality of image. In addition, input image dimensions can be scaled by varying the number of sub-streams. In SPPA, there is no need to store feature

TABLE 3
Comparison of SPPA with existing architectures

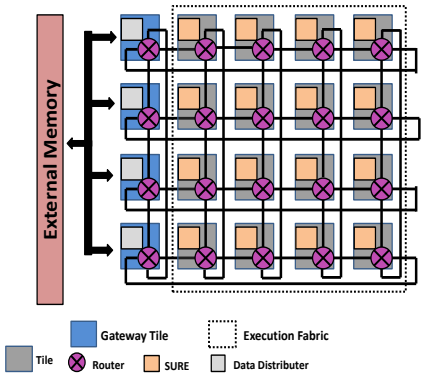
Architecture	**[26]**	**[27]**	**[29]**	**[30]**	**SPPA**
Face Detection involved	No	No	Yes	No	No
Feature Extraction	MPCA	WMPCA	PCA	PCA and MPCA	WMPCA
Classifier	NNC	NNC	NNC	NNC	RBFNN classifier
Training	Off-line	Off-line	Off-line	On-line	Off-line
Recognitions per second	91	26	45	19	450
Training database size	1000 images	5 classes, 3 samples per class	6 classes, 10 samples per class	2048 images	450 classes, samples per class not limited
Implementation platform	ALTERA EP20K600C B652C7	ALTERA EP20K200E FC484-2X	XILINX Virtex-5 FPGA	XILINX Virtex-4 XC4VFX12	XILINX Virtex-6 LX550T
Maximum operating frequency	91 MHz	33.33 MHz	Not mentioned	136.2 MHz	108 MHz
Supported number of PCs	20 PCs stored on set of registers	20 PCs stored on FPGA	59 PCs stored on Block RAMs	16 PCs stored on external memory	32 PCs stored on external memory
Face image dimension	64×64	Not mentioned	20×20	32×32	128×128

vectors for all the images in the database, instead we store only mean and variances of clusters. In the architectures using NNC, $RecPS$ depends on the number of training images in the database. Instead in SPPA, $RecPS$ depends on the number of classes used, i.e., real-time operation of SPPA is not constrained by the number of samples per class.

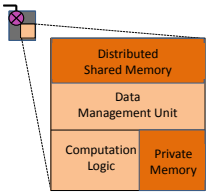
The SPPA for real-time FR is capable of performing FR on databases consisting of large number of classes with more than one cluster per class and images of large dimensions. SPPA with a real-time high frame rate face detector and a high speed image resize unit [39], is an ideal solution for real-time FR. We also observe that scalability of SPPA is limited by the external memory throughput. In order to assure better scalability and programmability, in the next section, we extend our design to a reconfigurable multi-core processor environment.

5 A SCALABLE AND RECONFIGURABLE MULTI-CORE ARCHITECTURE FOR FACE RECOGNITION

SPPA, typical of a dedicated architecture, requires rebuilding the circuit every time the specifications, such as $RecPS$, N_{class}, input image dimensions etc., are changed. Therefore we look for a reconfigurable solution which can process parallel streams and sub-streams of computations. We find that REDEFINE [6], a multi-core CGRA consisting of multiple Compute Elements (CE) connected over a Network on Chip [40], is a suitable candidate for this model. Modular processing in our FR algorithm exposes parallel and independent computations under each region. The software solution of our algorithm implemented on REDEFINE does not meet the real-time requirements, which necessitates hardware acceleration. To implement hardware acceleration in REDEFINE, each stream in SPPA, as shown by a dotted oval in Fig. 4, is realized as a processing core. We replace the general purpose CEs on REDEFINE with these domain specific programmable cores, termed Scalable Unit for Region Evaluation (SURE), to achieve real-time FR. Through reconfiguration, SURE cores serve as custom data paths for ESP, HNC, ONC and DU operations. We call this architecture as REDEFINE for Face Recognition using SURE Homogeneous cores (REFRESH). This reconfigurable multi-core processor environment enables introduction of a number of processing elements as hardware accelerators based on the need for scalability with respect to N_{class}, N_{clust} and image dimensions.



(a) Architecture of REFRESH



(b) Architecture of SURE

Fig. 9. Architectures of REFRESH and SURE

In section 5.1, we explain the architecture of REFRESH and internals of SURE. In section 5.2, we describe in detail architecture of computation logic in SURE. The intercommunication of SUREs is described in section 5.3. To achieve high throughput, each stream of SPPA is pipelined and consists of parallel sub-streams of computation in every pipeline stage. In the case of SURE, the limited network data width does not support supply of multiple data elements to SUREs over NoC. This restricts us from realizing multiple sub-streams of computation and pipeline stages in the SUREs. Therefore, we reuse the resources for ESP, HNC and ONC and use a single sub-stream of computation in each SURE to reduce resource utilization. To overcome the limitation due to NoC bandwidth, we exploit parallelism at the level of input images to achieve scalability in the design. In section 5.4 scalability of REFRESH is discussed in detail. Emulation and ASIC synthesis results are reported and discussed in section 5.5.

5.1 Architecture of REFRESH and SURE

The architecture of REDEFINE is explained in detail in the work by Mythri et al. [6]. REDEFINE architecture consists of a number of tiles connected over a NoC. Each tile in REDEFINE consists of a Compute Element (CE) for processing data and a router to accept, send or forward packets of instructions or data to CEs where they are destined to. Architecture of REFRESH consists of REDEFINE with the tiles containing CEs and FR specific SUREs to accelerate the computations for real-time FR. The tiles of REFRESH containing domain specific SUREs together constitute the execution fabric. A column of tiles termed Gateway tiles connect the execution fabric to the external memory. With SUREs as domain specific hardware accelerators at each tile of execution fabric, the task of each CE is to receive incoming data and send the same to the appropriate section of computation logic in SURE. We make this design further efficient by implementing these functionalities in a programmable Data Management Unit (DMU) of SURE and omitting the CEs from REFRESH as shown in Fig. 9(a). The DMU is made programmable to enable configuration of SUREs for different image dimensions, N_{class}, $N_{cluster}$ etc. Fig. 9(b) shows functional blocks in SURE. SURE consists of a programmable DMU, computation logic, a Distributed Shared Memory (DSM) and a private memory. Programmable units, termed data distributors, connected to the Gateway tiles perform the operation of distribution of data received from external memory to the tiles of corresponding row. The data received from data distributors over NoC by SURE is stored in corresponding DSMs. The DMU fetches the data to be processed from DSM and distributes the data to appropriate inputs of computation logic. The intermediate results of computation and look-up tables are stored in a private memory which is accessible only from the computation logic. Architecture of computation logic in SUREs is explained in detail in section 5.2.

5.2 Computation logic in SURE for FR

The computation logic in SURE is designed to perform ESP, HNC and ONC in a single region of input image. During final decision for classification, all the SUREs communicate among themselves for weighted addition of RBFNN outputs and find the maximum.

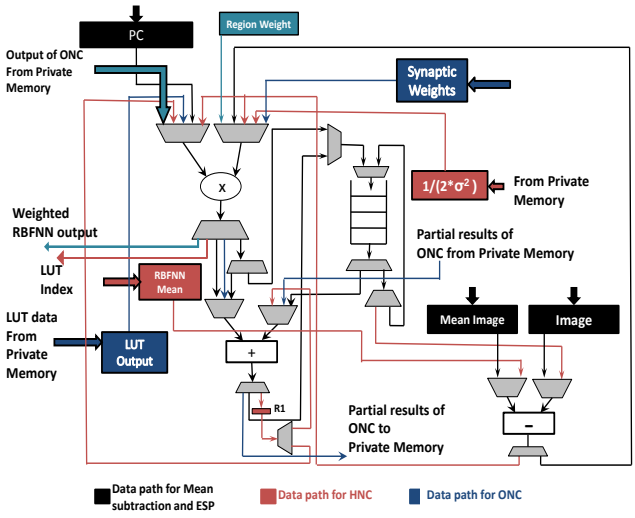


Fig. 10. Computation logic in SURE for Face Recognition

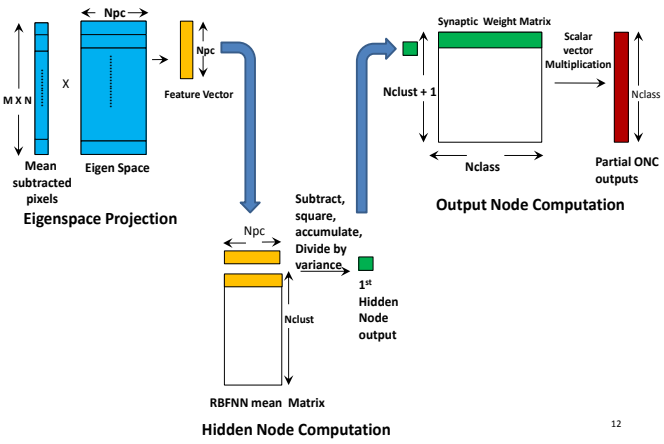


Fig. 11. Sequence of computations in SURE

The computation logic in each SURE is shown in Fig. 10. It consists of a multiplier, an adder, a subtracter and a circular FIFO in combination to perform all the operations in a region. Fig. 11 and Algorithm 1 describe the sequence of computations in SURE for ESP, HNC and ONC. Computations are sequenced in such a way that intermediate results are used for further computations without the need for local buffering and unnecessary write backs to memory. In addition, the sequence of computations is designed to achieve high spatial and temporal data localities. Mean image, Eigen space matrix, RBFNN mean and synaptic weights are pre-computed during off-line training and are stored in external memory. An input image to be recognized is written to a pre-determined location in the external memory. RBFNN variances, region weights and Look-up table for exponentiation computed during training are stored in the private memory.

For mean subtraction, the input image pixels and corresponding mean image pixels are fetched from external memory and subtracted to get mean subtracted image pixels. During ESP, the mean subtracted image needs to be multiplied with the eigenspace matrix stored in the external memory. Each output from the mean subtraction is multiplied with corresponding row of eigenspace matrix. The partial results are accumulated on a circular FIFO shown in Fig. 10. Thus, when all the input image pixels go through mean subtraction and ESP, extracted feature vector for the input image will be available in the circular FIFO.

We do not store the feature vector components back in private or external memory and hence avoid memory read and write cycles. For HNC, feature vector components in the circular FIFO are subtracted from a column of RBFNN mean matrix. For better data locality, we store the RBFNN mean matrix in transposed form in the external memory which enables row-wise access. After subtraction, each component is squared, and accumulated. For division by $2 \times \sigma^2$, a factor $\frac{1}{2 \times \sigma^2}$ is precomputed for each class and stored in private memory of corresponding SURE. Here σ is standard deviation of samples in corresponding cluster. Squared and accumulated result is multiplied with this factor and exponentiation is performed on the result using a lookup table stored in the private memory. The output of i^{th} hidden node is equal to the scalar output from computation on i^{th} row of RBFNN mean matrix. After computation of HNC output, for ONC it is

Algorithm 1: Computations in SURE for ESP, HNC and ONC

Input: Matrices $Image_{1\times L}$, $MeanImage_{1\times L}$, $EigenSpace_{L\times N_{pc}}$, $RBFMean_{N_{clust}\times N_{pc}}$, $RBFVar_{1\times N_{clust}}$, $RBFWeight_{(N_{clust}+1)\times N_{class}}$ and scalar $RegWeight$

Here $L = M \times N$;

Output: Matrix $ONCOut_{1\times N_{Class}}$

for $i = 1$ *to* L **do**
  $MeanSubPixel$ = $Image(1,i)$ - $MeanImage(1,i)$;
    // mean subtraction
  // Multiplication of $MeanSubPixel$ with a single row of $EigenSpace$. Partial results are accumulated in $FeatureVector$
  for $j = 1$ *to* N_{pc} **do**
    $FeatureVector(1,j)$ = $Feature_Vector(1,j)$ + $MeanSubPixel \times EigenSpace(i,j)$;
  end
end
for $i = 1$ *to* N_{clust} **do**
  // Subtraction of $FeatureVector$ from a row of $RBFMean$, squaring the elements and addition
  for $j = 1$ *to* N_{pc} **do**
    $SubSqSum$ = $SubSqSum$ + $(FeatureVector(1,j)$ - $RBFMean(i,j))^2$
  end
  // Multiplication of $SubSqSum$ by $\frac{1}{2\times\sigma^2}$ and exponentiation
  $HNCOut$ = $exp(\frac{SubSqSum}{2\times RBFVar(i)})$
  // Multiplication of $HNCOut$ by corresponding row of $RBFWeight$
  for $j = 1$ *to* N_{class} **do**
    $ONCOut(1,j)$ = $ONCOut(1,j)$ + $HNCOut \times RBFWeight(i,j)$
  end
end
// Bias input multiplication
for $i = 1$ *to* N_{class} **do**
  $ONCOut(1,i)$ = $ONCOut(1,i)$ + $RBFWeight(N_{cluster}+1,i)$
end

multiplied with a single row of weight matrix and intermediate results are stored in private memory and accumulated. These computations are repeated for the rest of HNC and ONC outputs. The number of clock cycles required to perform ESP, HNC and ONC in each region is equal to

$$T_{reg} = N_{pc} \times \left(\frac{MN}{N_{reg}}\right) + N_{pc} \times N_{clust} + (N_{clust}+1) \times N_{class} \quad (7)$$

The SUREs that operate independently on corresponding regions need to communicate among themselves for the final classification, which is described in section 5.3.

5.3 Connecting SUREs

For the FR algorithm described in [4], with N_{reg} equal to 16, we need 16 such SUREs connected over the NoC. After completion of ESP, HNC and ONC, each SURE holds RBFNN output vector for its region. During final classification, output vectors of all the regions are weighted and added. In this stage, adjacent SUREs communicate between themselves by sending the computed ONC outputs. Before sending, these ONC output components are multiplied by corresponding region weights. The data path shown in Fig. 9(b) includes computation logic for region weight multiplication. Fig. 12 shows transfer of ONC outputs between SUREs for final classification with description of steps involved. Finally the sum of RBFNN outputs of all the regions is available in a SURE where maximum among them is computed using a circuit for finding maximum. The circuit for finding maximum value is similar to the one shown in Fig. 6(b). In order to retain clarity in Fig. 12, we do not show implementation of the logic which finds the maximum value. In the REFRESH NoC, three clock cycles are consumed in forming the data packet in the routers and sending it. As this process is pipelined, to form and send X packets from a tile it takes $X + 3$ number of cycles. As this latency of three cycles is negligible compared to the number of packets being transferred, we ignore it in our analysis. Thus, the number of clock cycles required in exchange of ONC outputs

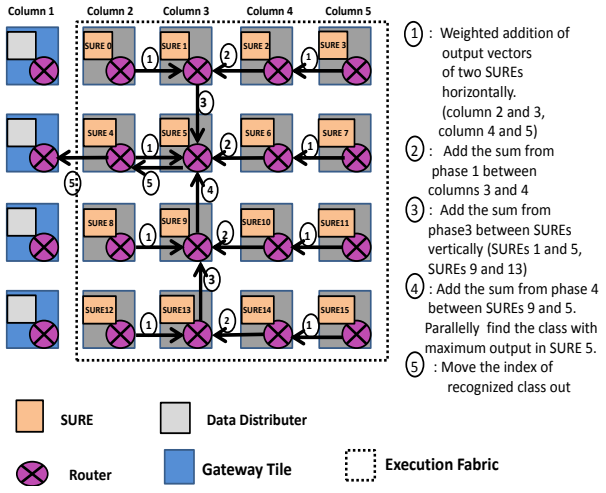


Fig. 12. Exchange of RBFNN output vector between SUREs for final classification

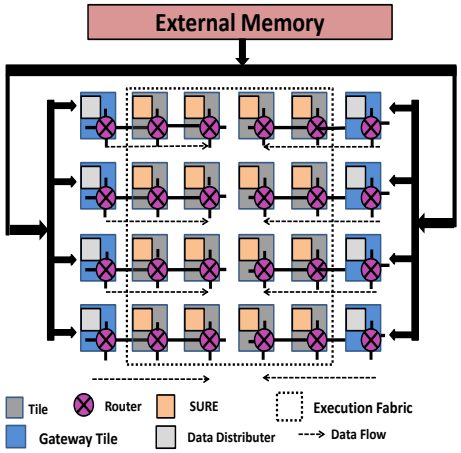


Fig. 13. Distribution of data from external memory among tiles of REFRESH

and computation of maximum, T_{am}, is equal to

$$T_{am} = log_2 N_{reg} \times N_{class} \quad (8)$$

For data in REFRESH, we use fixed point representation described in section 4.6. Among the data stored in external memory of REFRESH, highest number of bits i.e., 32 bits are used to represent RBFNN mean and synaptic weights. Equation 7 requires continuous supply of data for computation at each SURE. If the data width of NoC is kept at 32 bits, single data distributor in each row can provide 32 bits of data to each SURE every 4 cycles. As the supplied 32 bit data gets consumed in a single clock cycle, this data width of NoC keeps SUREs in waiting state for data most of the time. Increasing the data width to 128 bit can bring the average supply rate at each SURE to 32 bit per cycle. A better way of maintaining this supply rate is by introduction of two columns of gateway tiles with NoC data width of 64 as shown in figure 13. Data of 128 bits from external memory is divided among two gateway tiles. Each data distributor in gateway tile distributes data to two SUREs which makes availability of 32 bit data every cycle at each SURE possible. The additional column of gateway tiles help in reducing the NoC data width from 128 bits to 64 bits. Although this data distribution method restricts horizontal growth of REFRESH execution fabric, the scaling methods for REFRESH introduced in section 5.4 do not get affected. We name the set of 16 tiles in execution fabric shown in Fig. 13, that are impregnated with SUREs as Set_{16}. As there are four rows of tiles in a Set_{16}, we run the external memory at a clock frequency four times faster than the one used for execution fabric.

We fix $RecPS$ at 450 recognitions per second according to the design specification, image dimension at 128×128 and N_{pc} at 32. We find the number of classes that can be supported by a Set_{16} as follows.

$$T_{reg} + T_{am} = \frac{OperatingFrequency}{RecPS} \quad (9)$$

In section 5.4 we discuss scalability of REFRESH for supporting databases with larger number of classes and recognitions per second.

5.4 Scalability of REFRESH

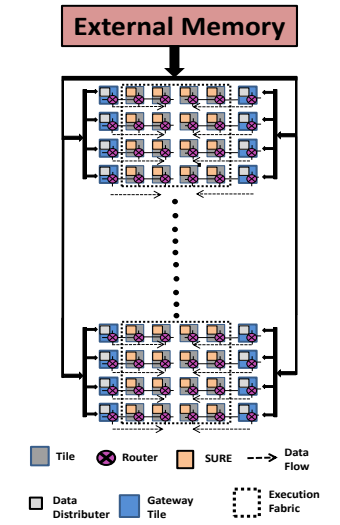
To increase the number of classes that can be supported by the FR module, we need to increase the computations that are performed in parallel. Increasing the number of arithmetic operators in SUREs does not increase the system throughput as it needs higher memory bandwidth to supply data to them in parallel. From equation 9 it is found that maximum number of classes the REFRESH can support is dependent on the $RecPS$. We observe that during recognition of different input images the data from external memory vary only in the input image pixels. Scalability in REFRESH is brought about by reusing the data fetched from external memory for processing multiple input images in parallel. For a given memory throughput, in order to support larger database, we suggest two ways of scaling the architecture.

- Single SURE Per Node (SSPN): There is one SURE implanted in each tile of REFRESH. The hardware structure is as shown in Fig. 14(a). In this structure we use multiple Set_{16}s connected in parallel. Each Set_{16} works on different input image. For a given $RecPS$, increasing the number of Set_{16}s effectively reduces the recognitions to be performed per Set_{16}. This in turn increases the available duration for processing each image per SURE, enabling accommodation of increased number of classes. If there are N_{SURE} number of SUREs in total, each Set_{16} performs $\frac{RecPS \times 16}{N_{SURE}}$ number of recognitions per second. ESP, HNC, ONC and final classification are performed in parallel on all the Set_{16}s. For SSPN, equation 9 is re-written as

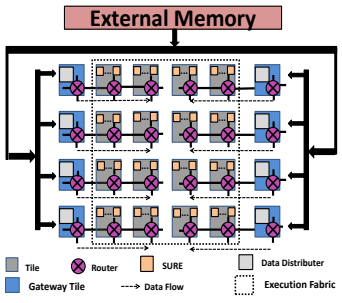
$$T_{reg} + T_{am} = \frac{OperatingFreq \times N_{SURE}}{RecPS \times 16} \quad (10)$$

- k-SURE Per Node (k-SPN): There are k ($k \geq 2$) SUREs implanted in each tile of REFRESH. In the SSPN set up, we need to architect network in such a way that the network grows as the number of Set_{16}s grow. To avoid this network growth, in k-SPN we reuse the data from external memory at tile level. As shown in Fig. 14(b), we implement a single Set_{16} and we implant more than one SURE under each tile. Each SURE under a tile works on a region of separate input image. Here too ESP, HNC and ONC are performed in parallel on SUREs. But the ONC output addition and classifications for different images are performed in sequence due to common network path. For k-SPN, equation 9 is re-written as

$$T_{reg} + \frac{N_{SURE}}{16} \times T_{am} = \frac{OperatingFreq \times N_{SURE}}{RecPS \times 16} \quad (11)$$



(a) Single SURE Per Node



(b) k-SURE Per Node

Fig. 14. Scalability of REFRESH

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2016.2539164, IEEE Transactions on Parallel and Distributed Systems

TABLE 4
Device Utilization of REFRESH on Virtex-6 LX550T FPGA ; $RecPS$=450, N_{SURE} = 16, M=128, N=128, N_{pc}=32, N_{class}=417 and N_{clust}=417. For N_{reg}=16

Number of slice registers	130766 out of 687360 (19%)
Number of slice LUTs	156390 out of 343680 (45%)
Number of Block RAMs	213 out of 632 (33 %)
Number of DSP48Es	176 out of 864 (20 %)

TABLE 5
ASIC synthesis results of REFRESH on Cadence RTL Compiler; $RecPS$=450, M=128, N=128, N_{pc}=32, N_{class}=417 and N_{clust}=417, N_{reg}=16 and N_{SURE} = 16

Maximum Operating Frequency	184 MHz
Area	5.7 mm^2
Power	1.22 W

In equation 10 and equation 11 we see that input image dimensions, N_{class} and N_{clust} are dependent on N_{SURE} and the rate at which SUREs operate. Although increasing the number of SUREs in REFRESH increases $RecPS$, the data supplied by external memory only increases by the additional image pixel data. Similar to equation 6, for SSPN and k-SPN we express the maximum data that can be supplied by the external memory to REFRESH as

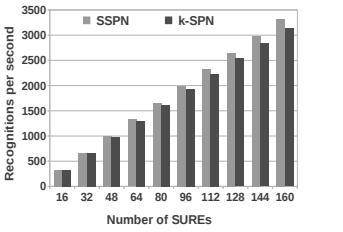
$$(M \times N \times ((B_i + B_{mean}) \times \frac{N_{SURE}}{16} + N_{pc} \times B_{pc}) + (N_{pc} \times N_{clust} \times B_m + (N_{clust} + 1) \times N_{class} \times B_w) \times 16) \times \frac{1}{2 \times Data\_width} = \frac{\beta \times Freq_{DDR} \times N_{SURE}}{RecPS \times 16} \quad (12)$$

In section 5.5 using synthesis results we show that the REFRESH can be scaled to support different input image dimensions, N_{class}, N_{clust} and $RecPS$. In addition we show the limitations introduced by the external memory throughput.

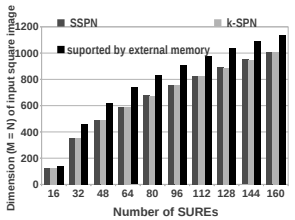
5.5 Emulation on FPGA and ASIC synthesis

For functional testing, we emulate a Set_{16} of REFRESH with single SURE per node on M503 module of Pico Computing systems [37]. We simulate and synthesize the design for Virtex 6 FPGA on Xilinx ISE. The device utilization is listed in Table 4. DDR3 SDRAM on M503 module has a data width of 64 and it can supply 128 bits of data every cycle operating at 400 MHz. From the synthesis report, the maximum operating frequency of REFRESH on FPGA is found to be 100 MHz. From equation 9, with β equal to 0.9, we get the number of classes that can be supported by REFRESH with SSPN format is equal to 417. In addition, we synthesized REFRESH at 65 nm technology on Cadence RTL Compiler. Table 5 lists the maximum operating frequency, power and area of the synthesized module.

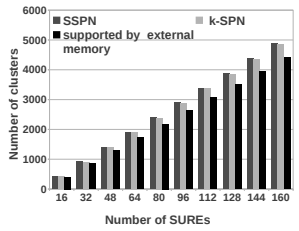
To scale REFRESH for larger databases, we replicate the SUREs multiple times either by SSPN or k-SPN as described in section 5.4. As the SUREs operate in parallel, operating frequency of REFRESH does not change with additional SUREs. The following plots in Fig. 15 and Fig. 16 and related analysis correspond to the synthesis results on FPGA. In Fig. 15(a) we plot $RecPS$ for SSPN and k-SPN with N_{class} equal to 417 using equation 10 and 11 respectively. Here we take N_{clust} equal to N_{class} similar to Fig. 2 and [4] for computational convenience. We observe that, with approximately same amount of data provided by external memory, the module is capable of performing higher recognitions per second with increase in N_{SURE}. Similarly we show scalability in input image dimensions with number of SUREs in Fig. 15(b). Recognition accuracy of REFRESH can be improved by having multiple clusters per class as described in section 3. In Fig. 15(c) we show the scaling in N_{clust} for a fixed number of classes and a given $RecPS$. We plot the maximum number of classes that can be processed by REFRESH for a given number of SUREs as shown in Fig. 15(d). In these plots we observe that by increasing N_{SURE} we enable REFRESH to operate on database of large sizes and to support high number of recognitions per second. Thus, REFRESH is able to achieve the desired scaling by reusing the data read from external memory. However, for a given N_{SURE}, the memory throughput of external memory limits the performance of REFRESH. To show this limitation, in Fig. 15(b), 15(c) and 15(d) we plot the maximum dimensions of input image,



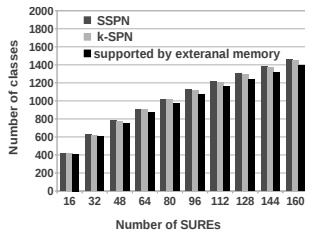
(a) Number of recognitions per second by REFRESH; M=128, N=128, N_{clust} = N_{class} = 417



(b) Maximum dimension (M = N) of input square image that can be processed by REFRESH and that can be supported by external memory; $RecPS$ = 450, N_{clust} = N_{class} = 417

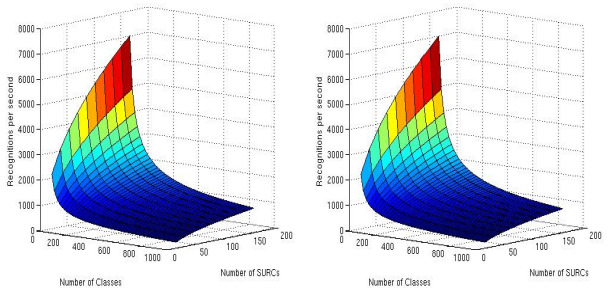


(c) Number of clusters that can be processed by REFRESH and that can be supported by external memory; $RecPS$ = 450, M=128, N=128, N_{class} = 417



(d) Number of classes that can be processed by REFRESH and that can be supported by external memory; $RecPS$ = 417, M=128, N=128, N_{clust} = N_{class}

Fig. 15. Scalability of REFRESH with N_{SURE}; N_{reg}=16, N_{pc} = 32



(a) SSPN (b) k-SPN

Fig. 16. Scalability of REFRESH with respect to N_{class}, $RecPS$ and N_{SURE}; N_{class}=N_{clust}, N_{pc} = 32, M = 128, N = 128 , N_{reg}=16

N_{clust} and N_{class} respectively supported by the external memory on M503 device. Three dimensional plots in Fig. 16(a) and Fig. 16(b) show variations in number of classes and recognitions per second, in the case of SSPN and k-SPN respectively, supported by REFRESH for different N_{SURE}.

REFRESH is capable of performing FR on databases of different sizes and at required rate of recognition. The number of recognitions per second and size of database can be scaled by introduction of additional SUREs. Thus, REFRESH is sufficiently flexible to be used in various practical conditions to meet real-time requirements due to its high accuracy algorithm and scalable architecture.

6 Conclusion

Real-time Face Recognition (FR) of subjects in crowd requires large number of recognitions to be performed every second. In addition, the algorithm used for FR needs to be sufficiently flexible to work in different practical conditions including illumination and pose variations of subjects in the captured images.

We introduced a modular algorithm for FR which is a combination of Weighted Modular Principle Component Analysis and Radial Basis Function Neural Network. The algorithm exhibited better recognition accuracy on images with illumination and pose variations than the algorithms used in existing hardware solutions for FR. In addition, due to inherent modularity, the algorithm provides scope for real-time hardware implementation. To meet the real-time requirements of FR, initially we proposed a Scalable Parallel Pipelined Architecture (SPPA). Parallel and independent computations in the introduced modular algorithm are performed as streams and sub-streams of computation. In order to support large databases, we store the coefficients in external memory. A novel data-layout in external memory minimizes the memory access latencies to achieve high memory throughput. Scalability is brought about by varying the number of sub-streams under each stream to support large databases and high rate of recognition.

For arbitrarily very large databases and large target recognitions per second, a fully dedicated architecture like SPPA will be not implementable due to very large problem size. Therefore, for better scalability and flexibility, we extended this work to a reconfigurable multi-core solution to achieve a highly scalable real-time FR. We came up with a programmable computation core, termed Scalable Unit for Region Evaluation (SURE), that performs parallel and independent computations in each stream of SPPA. We proposed an architecture, termed REDEFINE for Face Recognition using SURE Homogeneous cores (REFRESH), for real-time FR. REFRESH is made up of SUREs replicated and connected over an NoC. Computation sequences in SUREs are configured to achieve good spatial and temporal localities. We proposed two connection formats of SUREs to achieve scaling in REFRESH. Using these formats, REFRESH shows excellent scalability with respect to size of database and rate of recognition. With approximately same amount of data read from external memory, REFRESH shows comparable performance as that of SPPA. In addition, REFRESH can be further scaled with respect to input image dimensions, rate of recognition, number of classes and clusters by introduction of additional SUREs. The programmability of REFRESH helps to operate on databases of different sizes to support desired recognitions per second in a multi-core environment.

Due to considerably good recognition accuracy of the algorithm, impressive scaling ability and programmability, REFRESH is an ideal scalable multi-core solution for real-time FR.

Acknowledgments

The authors would like to thank Sukumar Bhattacharya for his valuable suggestions and guidance during algorithm exploration. The authors would also like to thank Batna A. Suryanarayana, Arnav Goel and Anup Kini for their contributions in the project.

References

[1] A. F. Abate, M. Nappi, D. Riccio, and G. Sabatino, "2d and 3d face recognition: A survey," *Pattern Recognition Letters*, vol. 28, no. 14, pp. 1885 – 1906, 2007, image: Information and Control. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865507000189

[2] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," *ACM Comput. Surv.*, vol. 35, no. 4, pp. 399–458, dec 2003. [Online]. Available: http://doi.acm.org/10.1145/954339.954342

[3] R. Jafri and H. R. Arabnia, "A survey of face recognition techniques." *JIPS*, vol. 5, no. 2, pp. 41–68, 2009. [Online]. Available: http://dblp.uni-trier.de/db/journals/jips/jips5.html# JafriA09

[4] G. Mahale, H. Mahale, A. Goel, S. Nandy, S. Bhattacharya, and R. Narayan, "Hardware solution for real-time face recognition," in *VLSI Design (VLSID), 2015 28th International Conference on*, Jan 2015, pp. 81–86.

[5] J. W. King, "Planning and design," in *Cisco IP Video Surveillance Design Guide*. CISCO, 2009. [Online]. Available: http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Video/IPVS /IPVS_DG/IPVS-DesignGuide.pdf

[6] M. Alle, K. Varadarajan, A. Fell, R. R. C., N. Joseph, S. Das, P. Biswas, J. Chetia, A. Rao, S. K. Nandy, and R. Narayan, "Redefine: Runtime reconfigurable polymorphic asic," *ACM Trans. Embed. Comput. Syst.*, vol. 9, no. 2, pp. 11:1–11:48, oct 2009. [Online]. Available: http://doi.acm.org/10.1145/1596543.1596545

[7] M. Turk and A. Pentland, "Eigenfaces for recognition," *J. Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, jan 1991. [Online]. Available: http://dx.doi.org/10.1162/jocn.1991.3.1.71

[8] A. M. Martínez and A. C. Kak, "Pca versus lda," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 2, pp. 228–233, feb 2001. [Online]. Available: http://dx.doi.org/10.1109/34.908974

[9] C. Garcia, G. Zikos, and G. Tziritas, "A wavelet-based framework for face recognition," in *WORKSHOP ON ADVANCES IN FACIAL IMAGE ANALYSIS AND RECOGNITION TECHNOLOGY*. Publications, 1998, pp. 84–92.

[10] R. Gottumukkal and V. K. Asari, "An improved face recognition technique based on modular pca approach," *Pattern Recogn. Lett.*, vol. 25, no. 4, pp. 429–436, mar 2004. [Online]. Available: http://dx.doi.org/10.1016/j.patrec.2003.11.005

[11] A. Kumar, S. Das, and V. Kamakoti, "Face recognition using weighted modular principle component analysis," in *Neural Information Processing*, ser. Lecture Notes in Computer Science, N. Pal, N. Kasabov, R. Mudi, S. Pal, and S. Parui, Eds. Springer Berlin Heidelberg, 2004, vol. 3316, pp. 362–367. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30499-9_55

[12] J. Moody and C. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, no. 2, pp. 281–294, June 1989.

[13] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Comput.*, vol. 3, no. 2, pp. 246–257, jun 1991. [Online]. Available: http://dx.doi.org/10.1162/neco.1991.3.2.246

[14] A. Howell and H. Buxton, "Learning identity with radial basis function networks," *Neurocomputing*, vol. 20, no. 13, pp. 15 – 34, 1998. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231298000162

[15] R. Debnath and H. Takahashi, "Learning capability: Classical rbf network vs. svm with gaussian kernel," in *Developments in Applied Artificial Intelligence*, ser. Lecture Notes in Computer Science, T. Hendtlass and M. Ali, Eds. Springer Berlin Heidelberg, 2002, vol. 2358, pp. 293–302. [Online]. Available: http://dx.doi.org/10.1007/3-540-48035-8_29

[16] S. Ranganath and K. Arun, "Face recognition using transform features and neural networks," *Pattern Recognition*, vol. 30, no. 10, pp. 1615 – 1622, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320396001847

[17] Y. W. Wong, K. P. Seng, and L.-M. Ang, "Radial basis function neural network with incremental learning for face recognition," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 41, no. 4, pp. 940–949, Aug 2011.

[18] G. Mahale, E. Bhatia, S. Nandy, and R. Narayan, "Vop: Architecture of a processor for vector operations in on-line learning of neural networks," in *VLSI Design (VLSID), 2016 29th International Conference on*, Jan 2016. [Online]. Available: http://cadl.iisc.ernet.in/cadlab/Vector_processor.pdf

[19] T. Subashini, V. Ramalingam, and S. Palanivel, "Breast mass classification based on cytological patterns using {RBFNN} and {SVM}," *Expert Systems with Applications*, vol. 36, no. 3, Part 1, pp. 5284 – 5290, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417408003886

[20] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, jul 2006. [Online]. Available: http://dx.doi.org/10.1162/neco.2006.18.7.1527

[21] J. W. King, "Large-scale fpga-based convolutional networks," in *Scaling up Machine Learning*. Cambridge University Press, 2011, pp. 399–419. [Online]. Available: ebooks.cambridge.org/chapter.jsf?bid=CBO9781139042918&cid=CBO9781139042918A158

[22] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 2, pp. 210–227, Feb 2009.

[23] A. V. Nefian, M. H. Hayes, and III, "Hidden markov models for face recognition," in *Proc. International Conf. on Acoustics, Speech and Signal Processing*, 1998, pp. 2721–2724.

[24] F. Yang and M. Paindavoine, "Implementation of an rbf neural network on embedded systems: Real-time face tracking and identity verification," *Trans. Neur. Netw.*, vol. 14, no. 5, pp. 1162–1175, sep 2003. [Online]. Available: http://dx.doi.org/10.1109/TNN.2003.816035

[25] F. Zuo and P. de With, "Real-time embedded face recognition for smart home," *Consumer Electronics, IEEE Transactions on*, vol. 51, no. 1, pp. 183–190, Feb 2005.

[26] R. Gottumukkal, H. T. Ngo, and V. K. Asari, "Multi-lane architecture for eigenface based real-time face recognition," *Microprocessors and Microsystems*, vol. 30, no. 4, pp. 216 – 224, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0141933105000645

[27] A. P. Kumar, V. Kamakoti, and S. Das, "System-on-programmable-chip implementation for on-line face recognition," *Pattern Recognition Letters*, vol. 28, no. 3, pp. 342 – 349, 2007, advances in Visual information Processing Special Issue of Pattern Recognition Letters on Advances in Visual Information Processing. (ICVGIP 2004). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865506000857

[28] S. Visakhasart and O. Chitsobhuk, "Multi-pipeline architecture for face recognition on fpga," in *Digital Image Processing, 2009 International Conference on*, March 2009, pp. 152–156.

[29] J. Matai, A. Irturk, and R. Kastner, "Design and implementation of an fpga-based real-time face recognition system," in *Field-Programmable Custom Computing Machines (FCCM), 2011 IEEE 19th Annual International Symposium on*, May 2011, pp. 97–100.

[30] N. Sudha, A. Mohan, and P. Meher, "A self-configurable systolic architecture for face recognition system based on principal component neural network," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 21, no. 8, pp. 1071–1084, Aug 2011.

[31] F. Samaria and A. Harter, "Parameterisation of a stochastic model for human face identification," in *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on*, Dec 1994, pp. 138–142.

[32] H. Wechsler, P. J. Phillips, V. Bruce, F. F. Soulie, and T. S. Huang, Eds., *Face Recognition : From Theory to Applications*, ser. NATO ASI. Springer, 1998, vol. 163.

[33] A. Georghiades, P. Belhumeur, and D. Kriegman, "From few to many: Illumination cone models for face recognition under variable lighting and pose," *IEEE Trans. Pattern Anal. Mach. Intelligence*, vol. 23, no. 6, pp. 643–660, 2001.

[34] A. M. Martinez and R. Benavente, "The AR Face Database," CVC, Tech. Rep., jun 1998.

[35] D. Wang, X.-J. Zeng, and J. A. Keane, "A clustering algorithm for radial basis function neural network initialization," *Neurocomputing*, vol. 77, no. 1, pp. 144 – 155, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231211005200

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

[37] "Hpc modules: Pico computing." [Online]. Available: http://picocomputing.com/products/hpc-modules/

[38] "Bluespec." [Online]. Available: http://www.bluespec.com/

[39] G. Mahale, H. Mahale, R. Parimi, S. Nandy, and S. Bhattacharya, "Hardware architecture of bi-cubic convolution interpolation for real-time image scaling," in *Field-Programmable Technology (FPT), 2014 International Conference on*, Dec 2014, pp. 264–267.

[40] A. Fell, P. Biswas, J. Chetia, S. Nandy, and R. Narayan, "Generic routing rules and a scalable access enhancement for the network-on-chip reconnect," in *SOC Conference, 2009. SOCC 2009. IEEE International*, Sept 2009, pp. 251–254.

Gopinath Mahale Gopinath Mahale is a research scholar at the Indian Institute of Science (IISc). He received his B.E. from Vishweswariah Technological Univeristy and MTech in Digital Systems from College of Engineering, Pune. He is associated with CAD Laboratory, IISc as a PhD student since August 2011. His research interests include digital system design, image processing and machine learning.

Hamsika Mahale Hamsika Mahale is a project associate at CAD Laboratory, Indian Institute of Science. She has received her MTech in Digital Systems from College of Engineering Pune. She is currently working on algorithms and architectures for Face detection and recognition system.

S.K. Nandy S.K. Nandy is a Professor at the Supercomputer Education and Research Centre, Indian Institute of Science,Bangalore. His research interests are in the areas of Low Power and High Performance Embedded Systems on a Chip, VLSI architectures for Reconfigurable Systems on Chip, and Architectures and Compiling Techniques for Heterogeneous Many Core Systems. Nandy received the B.Sc (Hons.) Physics degree from the Indian Institute of Technology, Kharagpur, India, in 1977. He obtained the BE (Hons) degree in Electronics and Communication in 1980, MSc (Engg.) degree in Computer Science and Engineering in 1986 and the Ph.D. degree in Computer Science and Engineering in 1989 from the Indian Institute of Science, Bangalore. He has over 150 publications in International Journals, and Proceedings of International Conferences.

Ranjani Narayan Dr. Ranjani Narayan has over 15 years experience at IISc and 9 years at Hewlett Packard. She has vast work experience in a variety of fields computer architecture, operating systems, and special purpose systems. She has also worked in the Technical University of Delft, The Netherlands, and Massachusetts Institute of Technology, Cambridge, USA. During her tenure at HP, she worked on various areas in operating systems and hardware monitoring and diagnostics systems. She has numerous research publications.She is currently Chief Technology Officer at Morphing Machines Pvt. Ltd, Bangalore, India.