Sigma J Eng & Nat Sci 34 (3), 2016, 393-405

Sigma Journal of Engineering and Natural Sciences
Sigma Mühendislik ve Fen Bilimleri Dergisi

Research Article / Araştırma Makalesi

ACCELERATING HANDWRITTEN SIGNATURE RECOGNITION USING INTELLIGENT ALGORITHM BASED EMBEDDED SYSTEM

Ali Rıza YILMAZ[1], Burcu ERKMEN*[1], Oğuzhan YAVUZ[2]

[1]Yildiz Technical University, Department of Electronics and Communication Engineering, Istanbul-TURKEY
[2]İstanbul Technical University, Istanbul-TURKEY

Received/Geliş: 22.02.2016 Revised/Düzeltme: 03.06.2016 Accepted/Kabul: 25.06.2016

ABSTRACT

In this work, intelligent algorithms designed on embedded hardware for signature recognition is presented. Feed forward Conic Section Function Neural Network (CSFNN) and Differential Evaluation Algorithm (DEA) are implemented on the Field Programmable Gate Arrays (FPGAs). Unified robust classifier CSFNN is applied on the preprocessed signatures for recognition purpose. DEA is used for training CSFNN in order to overcome local minimum problems. The implemented CSFNN on FPGA has the characteristics of flexible adaptable size providing various datasets. The CSFNN implementation on FPGA is realized using the 16-bit floating point arithmetic IEEE 754-2008 standard. The proposed on-chip CSFNN based signature recognition system described in VHDL has been implemented and evaluated on a high–end Virtex 7 -VC707 platform. The intelligent system embedded on FPGA is approximately 10^5 times faster than its equivalent software implementation.

Keywords: Conic section function neural network, FPGAs, signature recognition, differential evaluation algorithm.

AKILLI ALGORİTMA TABANLI GÖMÜLÜ SİSTEM KULLANARAK İMZA TANIMA İŞLEMİNİN HIZLANDIRILMASI

ÖZ

Bu çalışmada, imza tanıma işlemi için akıllı algoritmaların gömülü donanım üzerinde tasarımı sunulmuştur. İleri beslemeli Konik Kesit Fonksiyonlu Sinir Ağları (CSFNN) ve Diferansiyel Gelişim Algoritması (DEA), Sahada Programlanabilir Kapı Dizileri (FPGA) üzerinde gerçeklenmiştir. Birleştirilmiş ağ yapısına ve başarılı sınıflama performansına sahip CSFNN, tanıma amaçlı ön işlemlerden geçirilmiş imza verilerine uygulanmıştır. DEA, CSFNN'in eğitimi esnasında local minimum problemlerinin üstesinden gelebilmek amacıyla kullanılmıştır. FPGA üzerinde gerçeklenen CSFNN, çeşitli veri tabanlarının uygulanmasını sağlamak üzere esnek uyarlanabilir boyutlu bir karakteristiğe sahiptir. 16 bit kayan noktalı aritmetik kullanılarak IEEE 754-2008 standartında FPGA üzerinde CSFNN gerçeklenmiştir. Önerilen CSFNN tabanlı imza tanıma sistemi, çip üzerinde VHDL Donanım Tanımlama dili kullanılarak tanımlanmış, yüksek seviye Virtex 7 -VC707 platformu üzerinde çalıştırılmıştır. FPGA üzerinde oluşturulan Akıllı Sistem, karşılık düşen yazılım uygulamasına göre yaklaşık olarak 10^5 kat daha hızlı yanıt vermektedir.

Anahtar Sözcükler: Konik kesit fonsiyonlu sinir ağları, FPGA, imza tanıma, farksal gelişim algoritması.

[*] Corresponding Author/Sorumlu Yazar: e-mail/e-ileti: bkapan@yildiz.edu.tr, tel: (212) 383 59 18

1. INTRODUCTION

Signatures every day are used to authorize, some bureaucratic transactions, contracts, and to validate financial process and credit card receipt. The visual appearance of our signatures for verification and security purposes is especially taken into account by financial and commercial organizations. A biometric signatures recognition system is realized for recognizing the owners of signatures. In literature, many successful studies are presented for solving the handwritten signature recognition and verification problems. The aim of those works is to detect the forgeries using the signature verification methods. Two methods of the signature recognition and verification are available, on-line and off-line. The on-line method with a special device such as electronic tablet, pressure sensitive pens, and glove-based systems measures the sequential data, such as handwriting speed and pen pressure [1-2]. The off-line method uses an optical scanner to obtain handwriting data from a signature written on paper. For the off-line method, two main approaches are used. One of them is pseudo-dynamic approach, the other one is static approach that uses global [3] or geometric and topological features and grid information [4].

In recognition process, the features of the test signatures are compared with stored signature database previously. NNs have been robust classifiers to widely used for automatic signature verification for a long time [5]. There are many NNs that are used for the classification of signatures or hand-written recently. One of them is multi-layer perceptron (MLP) which is used for on-line and off-line hand-written recognition problems [6]. Moreover, radial basis function (RBF), back-propagation NNs (BPNs) [7], and Conic Section Function Neural Network (CSFNN) [4, 8] were used as robust classifiers for signature recognition. CSFNN training using back-propagation algorithm could be suffer from stucking into local minimums which can be solved by using repeated trainings. Although the gradient algorithms are based on local search methods, DEA relies on global search method to avoid local minimum problem [9].

The majority of numerous implementations of Artificial Neural Networks (ANNs) were realized as software platform on sequential processors. However, NN models require a lot of computing time to be simulated on a sequential machine that results in a great difficulty to investigate the behavior of large NNs and to verify their ability to solve problems [4]. On the other hand, ANN hardware implementation with ASIC provides compact and high speed custom designed circuits due to the use of hardware architectures matching the parallel structure of ANNs. However, while developing ANN sub-circuits, ASIC device realization is time consuming, expensive and inflexible [4]. Powerful alternative to VLSI realizations in terms of low cost and re-configurability, ANNs were realized on the reconfigurable platforms such as digital signal processing (DSP), and Field Programmable Gate Array (FPGA) based implementations. FPGAs with their massively parallel platform are preferable for the hardware implementations of ANNs. The main benefits of the usage of a system-on-programmable-chip implementation are re-configurability and low cost design provided by the high level of parallel integration. The vast majority FPGA implementation of ANN architectures, the learning algorithms are static implementations for off-chip applications without learning capability [10]. One of the off-chip pattern classification applications using CSFNN implemented on low cost FPGA platform was presented in [11]. Some of FPGA implementations, run time reconfiguration was achieved for online applications [12-14]. FPGA realizations of MLP with Levenberg-Marquardt Algorithm [12], the back-propagation algorithm [13-14] and DEA [15] have been developed as on-chip methodology. Due to the advantages of FPGA, the systems which operate the complex the hand-written and the signatures data have been realized recently [16, 17].

The main objective of this paper is to accelerate intelligent system performance on FPGA platform using on chip learning technique. Highly nonlinear signature recognition problem was performed with CSFNN using DEA training method on embedded hardware.

2. SIGNATURE RECOGNITION SYSTEM

Data acquisition, signature preprocessing, classification, performance evaluation are the main stages of typical Signature Recognition System. CSFNN Based Signature Recognition System includes two main phase, namely, the training and the recognition phases. The performance of pre-processed signature databases was evaluated in Signature within the scope of this work. Each signature database divided into two parts each containing the training and the test samples before applied to intelligent system. The training samples were applied to CSFNN in the training phase where the network parameters were updated to minimize the error signals using an evolutionary algorithm. After the training phase was completed, the test samples were presented to CSFNN feed-forward module in recognition phase as shown in Figure 1.

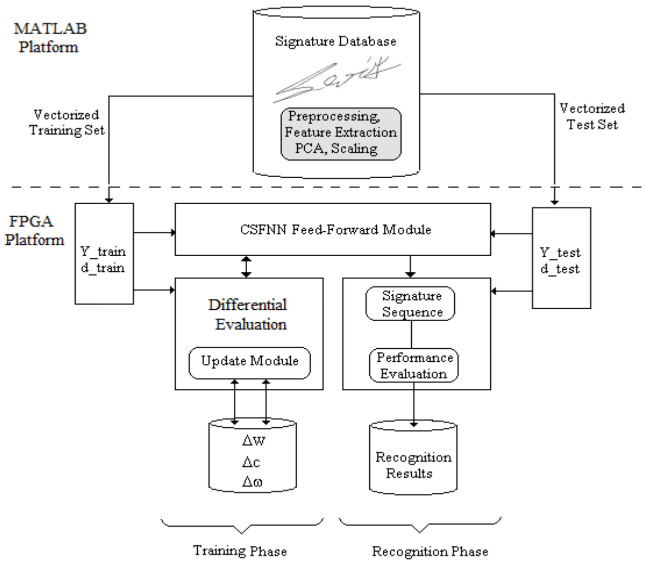

Figure 1. Data flow within signature recognition system

2.1. Data Acquisition and Preprocessing

The performance of Signature Recognition System has been evaluated using Signature Dataset from [4] in this work. In the databases, the signature samples, collected from various people, were processed using several techniques before applying them into CSFNN. The following techniques examined in [18]. These methods are summarized briefly as follows: the noise reduction algorithm, the masking, and the signature skeletonization were realized on the signature images acquired by scanner. After applying preprocessing techniques, the feature vectors were constituted by using the feature extraction methods. The feature numbers of high dimensional datasets including grid information and global features, which was not suitable for presenting to CSFNN as it stands, were decreased, namely compressed, by Principle Component Analysis (PCA), linear scaling and selecting centers were the last steps to form the database.

2.2. Signature Database

Signature Dataset from [18] contains 256 signature images from 8 people with 32 signatures for each person. The database comprises of 10 features after PCA. This database contains 8 classes where each class refers to a signature owner. The training and test samples were chosen from signature databases randomly. 200 signature samples including 25 samples for each class form the training set and the rest have been used for testing.

2.3. The Feed-Forward Network

The CSFNN is a layered fully connected feed-forward network which combines the propagation rules of RBFs and MLPs on a single NN with a unique propagation rule that make possible simultaneous use of advantages of both networks [19]. Ellipses, hyperbolas or parabolas in between hyper-plane and hyper-sphere, special cases of CSFNN, are all valid for decision regions of CSFNN. Automatic decision boundaries are provided through the distribution of a given dataset [4, 20]. It makes satisfactory classification performance even if high dimensional databases are applied to CSFNN [4, 8, 21]. The CSFNN with 10 inputs, 10 hidden neurons and 3 output neurons has been designed for this signature recognition problem. The neural computation is different in the hidden neurons and the output neurons in CSFNN as shown in Figure 2. The algorithm including the feed forward propagation, the updating and the testing processes were expressed as pseudo code in the Figure 3 and Figure 4-5. The widely used notations for CSFNN description were given in Table 1.

Table 1. The notations or CSFNN description

w_{ij}	weights for connection between i. input node and j. hidden neuron
w_{jk}	weights for connection between j. hidden neuron k. output neuron
ω_j	opening angle for j. hidden neuron
c_{ij}	center coordinates for i. İnput node and j. hidden neuron
X	input in the vectorized form
a_j	neuron output for j. hidden neuron
a_k	neuron output for k. output neuron
d_train	target values for training set

The pseudo code of the feed-forward algorithm for CSFNN training is shown in Figure 3. In the beginning of the algorithm, weights, centers, opening angles are set to their initial values (Lines 2-7). After initialization, the algorithm begins to process each sample in dataset. The propagation rule of hidden neurons can be derived using the analytical equation "consec" (Line 11). The output neurons are inner product type. The outputs of CSFNN are produced at Line 19. The bipolar sigmoid functions represented as "act_func" in the algorithm are used as derivative transfer functions that compress an infinite input range to a finite output range [-1,1]. The mathematical models of these functions are given in Table 2.

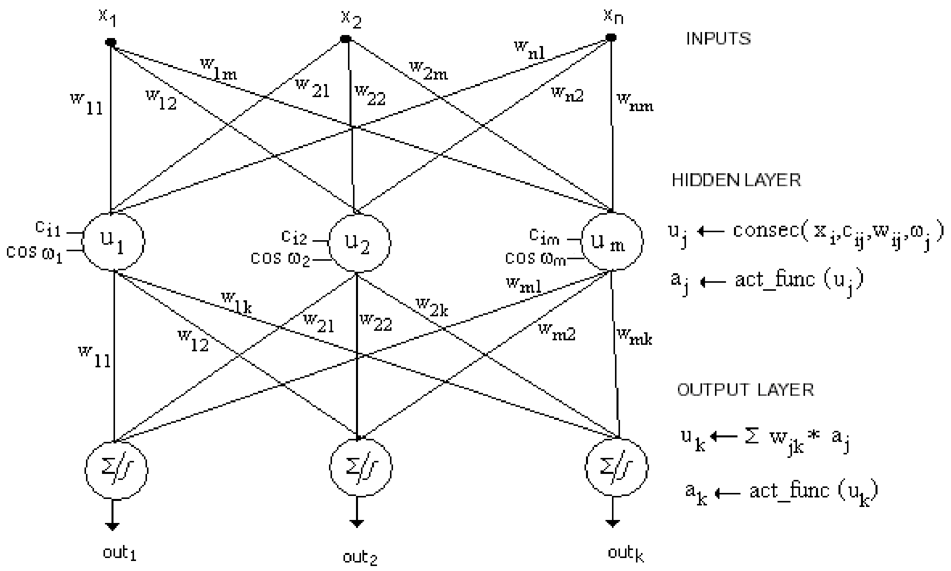

Figure 2. CSFNN structure

Algorithm 1 Feed-Forward Algorithm of CSFNN for Training

```
1:  /* Set initial values of weight, the center and the opening angle
2:  $w_{ij} \leftarrow$ rand{-0.5 , 0.5}
3:  $w_{jk} \leftarrow$ rand{-0.5 , 0.5}
4:  $c_{ij} \leftarrow$ rand {$x_i$} for all $x \in$ Train_Set
5:  if initial = MLP then $\omega_j \leftarrow \pi/2$
6:  else if initial = RBF then $\omega_j \leftarrow \pi/4$
7:  end if
8:
9:  for ep_num=1 to Epoch_Number do
10:     for tr=1 to Train_Set_Size do
11:         $u_j \leftarrow$ consec($x_i$, $c_{ij}$, $w_{ij}$, $\omega_j$)
12:         $a_j \leftarrow$ act_func ($u_j$)  for all Hidden_Neurons
13:         $u_k \leftarrow w_{jk} * a_j$
14:         $a_k \leftarrow$ act_func ($u_k$) for all Output_Neurons
15:         /* $a_k$ utilized in Algorithm 2
16:     end for
17: end for
18: for all Train_Set do
19:         Y_train $\leftarrow a_k$
20: end for
```

Figure 3. Pseudo code of feed-forward algorithm of CSFNN

Table 2. Function sets in CSFNN algorithm

Function Name	Function Legend	Mathematical Model
Consec	Conic Section Function for Hidden Neuron Model	$u_j^p(x) = \sum_{i=1}^{n} (x_i^p - c_{ij}) w_{ij} - \sqrt{\sum_{i=1}^{n} (x_i^p - c_{ij})^2} \cos \omega_j$
act_func	Bipolar Sigmoid Activation Function	$f_j^p(u) = \frac{2}{1 + e^{-2 \cdot u_j^p}} - 1$
Rand	Random Entries generator	Matlab Default Func.

2.4. The Training Algorithm

The training is performed using differential evaluation algorithm [22] after every forward pass of the network. Algorithm 1 in Figure 3 and Algorithm 2 in Figure 4 are run sequentially until all input vectors perform in the training set and the iterations reach its maximum epoch value which is represented as Epoch number.

DEA is the one of the heuristic algorithms for global parameter optimization. The derivative free optimization is obtained by DEA. The parameters of DEA are given in Table 3. Population size (NP), crossover rate (CR) and the scale factor (F) should be determined by user for minimizing the cost function. The main steps of this algorithm are the mutation, the crossover and the selection in basic pseudo code for in Figure4. Before the training of CSFNN with DEA, the solution set has been produced randomly between upper and lower limits (Line 1-2). Each solution set is called as a chromosome and the chromosomes generate the population. In addition, each variable of chromosomes is called as gene. CSFNN has m×(k+n) $m(k+n)$ weights, m×ncenter and m angle values to be optimized for n inputs, m hidden neurons and k output neurons.

According to DEA, firstly, the initial population has been applied to CSFNN architecture to obtain the best solution set which minimizes the cost function (Line 3-5). Then all population has been updated by applying mutation, crossover and selection processes (Line 6-18). In the selection process, the best chromosome is updated (Line 14-17). In this work, the epoch number has been determined as the stopping criterion. The cost function is defined by average squared error [23] given in Equation (1).

$$\varepsilon_{av} = \frac{1}{2N} \sum_{p=1}^{N} \sum_{i=1}^{k} \left[(d_i - Y_i)^2 \right] \tag{1}$$

Table 3. Notations for DEA description

Symbol	Quantity
NP	Population Size (number of the chromosomes) NP ≥ 4 (1, 2, 3, …, i)
D	Number of the Variable (Gene) (1, 2, 3, , j)
CR	Crossover Control Parameter [0,1]
G	Maximum Generation (1, 2, 3, …, Gmax)
F	Mutation Scale Factor [0,2]
$x_{j,I,G}$	The j. parameter of i. chromosome in G. generation. (Gene)
v_i	Mutant Vector
u_i	Trial Vector
r_i	The chromosomes which are chosen randomly to produce new generation *i*={1, 2, …, NP} ; $r_1 \neq r_2 \neq r_{3...} \neq r_{NP}$
$x^{(u)}$ $x^{(l)}$	Upper and lower limit values of variables

```
Algorithm 2 DEA for training CSFNN
1:  /* Initial population
    $\forall I \leq N_p \square \ \forall j \leq D$: $x_{J,I,G=0} = x_j^{(l)} + rand_j\,[0,1].(x_j^{(u)} - x_j^{(l)})$
2:  /* Find the best X:
3:  if $f_{min} = f(x_j)$ then $x_{best} = x_j$
4:  end if
5:  for i=0 to iteration number
6:   begin
7:  /* Mutation :
8:     $\forall j \leq D$: $U_{J,I,G=0} = x_{best} + F(x_{j,r1,G} - x_{j,r2,G})$
9:  /* Crossover :
10:     if $rand[0,1] \leq CR \square j = j_{rand}$ then $\forall v_{j,i} = u_{j,I,G}$
11:     else $\forall v_{j,i} = x_{j,I,G}$
12:     end if
13: /* Selection
14:     if $f(u_{j,I,G}) \leq f(x_{j,I,G})$ then $x_{I,G+1} = u_{I,G}$
15:    else $x_{I,G+1} = x_{I,G+1}$
16:    end if
17:  end for
18: end
```

Figure 4. Pseudo code of DEA for CSFNN training

2.5. The Testing

The testing is performed for all input vectors in testing set. Outputs of the network (Line 11) are produced according to last updated network parameters generated in Algorithm 2 which are set to the fix value during feed forward computation. Pseudo code of this feed-forward algorithm of CSFNN for testing is shown Figure 5.

```
Algorithm 3 Feed-Forward Algorithm of CSFNN for Testing
1:  / * Final values ($c_{ij}$,$w_{ij}$,$w_{jk,,}$$\omega_j$) generated from Algorithm1,2
2:    for  i=1 to Test_Set_Size do
3:        $u_j \leftarrow$ consec($x_i$,$c_{ij}$,$w_{ij}$, $\omega_j$)
4:        for all  Hidden_Neurons do
5:              $a_j \leftarrow$ act_func ($u_j$)
6:     end for
7:              $u_k \leftarrow w_{jk} * a_j$
8:              $a_k \leftarrow$ act_func ($u_k$) for all  Output_Neurons
9:        end for
10:  for all Test_Set  do
11:             Y_test $\leftarrow a_k$
12:  end for
```

Figure 5. Pseudo-code of feed-forward algorithm of CSFNN for testing

3. HARDWARE IMPLEMENTATION

The recognizer part of CSFNN based Signature Recognition System was implemented on hardware to achieve high level inherent parallelism of neural networks. The hardware based NNs were allowed to converge at a higher speed [4, 24] and to obtain a higher processing throughput [25] than software-based counterparts. In [4], the hardware architecture of CSFNN feed-forward computation was realized on mixed mode VLSI circuitry using chip in the loop technique. Alternatively to CMOS circuitry in [4], FPGA based CSFNN architecture perform on-chip training using DEA.

CSFNN architecture of Signature Recognition System has been implemented on FPGA (Virtex 7 VC707). Highly dense and inherent parallelism of FPGA implementation make possible to realize large-scale neural networks with a great numbers of arithmetic units in hardware. The implementation is programmed via VHSIC Hardware Description Language (VHDL) with fully synthesizable code. Mega-functions, IP cores and other libraries weren't utilized while implementing CSFNN architectures on FPGA. In order to program VHDL code onto the FPGA platform, Xilinx ISE Design Suite tool was used.

On-chip learning technique [4, 11] is used for improving performance of intelligent recognition system. The training takes place entirely on the chip. Only the inputs of the training dataset are supplied to the chip. The synaptic weight, center and angle values are updated according to DEA.

The training datasets are used for training the network and the test datasets are applied to CSFNN based FPGA platform to evaluate hardware's ability to generalize. Considering functional expressions of CSFNN given in the second section, the feed-forward algorithm consists of some arithmetic units and nonlinear functions. On FPGA implementation, addition, subtraction, multiplier and squarer units were realized using 16-bit floating point (FP) IEEE 754-2008 standard. FP cores for designing arithmetic modules have been developed during the work.

The propagation rule for CSFNN hidden layer is referred as "consec" function presented in Table 2. Accordingly this function, data flow and the arithmetic units within a CSFNN hidden neuron is demonstrated as a block diagram in Figure 6. This representation is converted to the propagation rule for output layer by setting the center coordinates and cosine of opening angles to zero. Therefore, the hidden neuron could be substitute for the output neuron throughout design. The hidden neuron realization, we call a primary neuron, constitute the fully connected CSFNN. A primary neuron could be assigned as a hidden neuron or an output neuron depending on neural network size determined by distribution of a given dataset. During the neuron type assignment procedure, the hardware architecture of a neuron model is controlled by input and output control units.

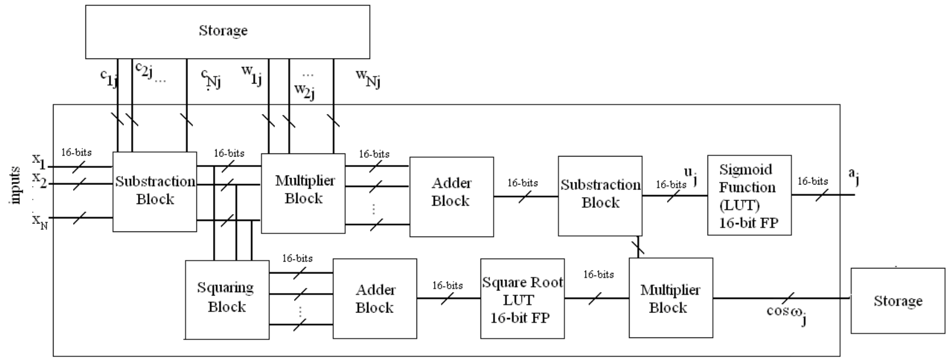

Figure 6. The primary neuron structure

Designing the hardware of a primary neuron, the FP arithmetic was realized for these computational units whereas the Look-Up Table (LUT) was used for the non-linear function realization such as bipolar sigmoid and the square-root functions. The sigmoid function was implemented using LUT which consists of 5019 samples which need 76 Kbits memory. Since the sigmoid function has the symmetric characteristic, the FP data in the LUT were stored for the right-hand side of the sigmoid function. The negative values of the function were obtained from the values by adding the sign bit. Therefore, the LUT needs less memory. As shown in Figure 6, CSFNN neuron in the hidden layer involves a square-root function which also demonstrates non-linearity characteristic implemented using LUT with 11882 samples. The function limited [0, 5] were segmented into variable resolution while LUT implementation to reach a maximally precise approximation [13]. In order to realize output neuron using primary neuron, the center coordinates and cosine of opening angles was set to zero.

The input control unit assigns simultaneously the signature dataset for every neuron of the layer in flexible adaptable sized CSFNN hardware. In this implementation, the clock signal (200 MHz) is drawn from the board. In order to realize testing process, the signature dataset was send to the CSFNN implementation via the serial port interface and the output control unit interprets the outputs and provides control signals to classify the outputs. The different sized CSFNN architectures can be built on the embedded device to estimate hardware demands in terms of the speed and the resource usage.

DEA is implemented on FPGA board in order to update CSFNN parameters during training. Table 4 presents the utilization of the FPGA resource for sub modules of DEA and feed forward CSFNN architectures. Crossover, mutation, selection, comparison and ROM units within DEA are demonstrated as a block diagram in Figure 7.

Table 4. The resource usage for the sub-modules of DEA and CSFNN

System Resource	Combinational ALUTs	Dedicated Logic Regiters	DSP 18-bit Elements
Initialization Module – DEA	--	845 -- 1%	--
Comparison Module – DEA	14759 -- 4.5%	2177 – 1%	1
Mutation Module – DEA	46012 -- 15%	--	130
Crossover Module – DEA	2083 -- 1%	2084 – 1%	--
Selection Module- DEA	2204 -- 1%	2217 – 1%	2
CSFNN	14264 -- 4.5%	1956 – 1%	1

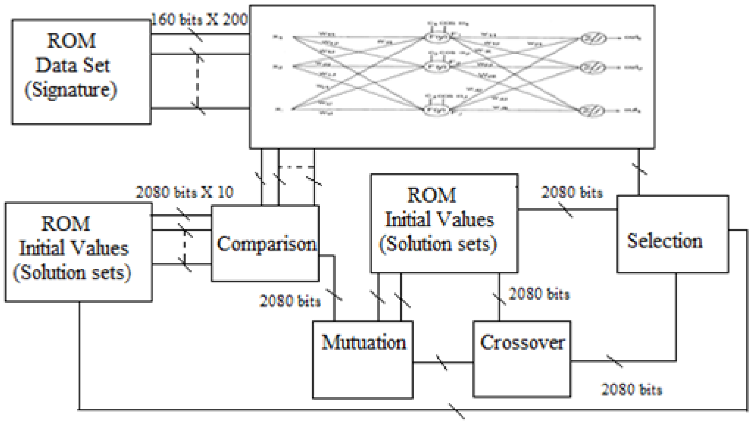

Figure 7. DEA structure for training CSFNN

4. SIGNATURE RECOGNITION PERFORMANCE

The signature recognizing performance of FPGA based system has been evaluated on Signature Dataset in terms of system resource utilization, speed, recognition accuracy. The proposed on-chip FPGA based hardware system was compared with a pure software based approach considering evaluation criteria.

4.1. Performance Evaluation

The overall signature recognition system comprises of MATLAB, FPGA platform and industry – standard peripheral interfacing unit. The signature recognition process is split into the training and the recognition phases. The training and test set consist of 200 signature samples including 25 samples for each class and 56 test samples including 7 samples, respectively. The CSFNN with 10 inputs, 10 hidden and 3 output neurons has 130 weights, 100 center values and 10 angle values to be optimized by DEA. In other words, each chromosome in DEA has 240 genes. During CSFNN parameters optimization process, the control parameters of DEA are set as in Table 5.

Table 5. The control parameters of DEA for training CSFNN

NP	F	CR	Epoch Number
20	0.6	0.6	3000 (Software) 150 (Hardware)

Feed-forward computation and training were implemented on the target FPGA board after synthesis and place and route process. At the recognition phase, the test signature samples were applied to the CSFNN based signature recognition system via the serial port interface while the updated weight, opening angle and center values were stored on distributed-memory of FPGA platform.

4.2. Performance Comparison

The signature recognition accuracy performance of CSFNN based FPGA implementation was compared with pure software accuracy performance [4]. The aim of the performance comparison is to reveal the weaknesses and advantages of the hardware implementations over software platform. Table 6 compare the accuracy performances of CSFNN – DEA based FPGA implementation with the performances of pure software based system running on CSFNN algorithm for highly nonlinear Signature Dataset. On chip learning based FPGA implementation recognized 180 training signature samples correctly among 200 training samples and 46 signatures correctly among 56 test samples. Classification boundaries were defined such that if the output is less than 0.5, it is assigned to 0.1 and if it is equal and more than 0.5, it is assigned to 0.9 during evaluation process. The feed forward neural network and training algorithm implemented on the FPGA doesn't show the same performance to that simulated on the PC using MATLAB. Due to process workload during iterative training, even small quantization errors increase cumulatively for some input samples.

Table 6. Signature recognition accuracies

FPGA		Matlab	
Train (%)	Test (%)	Train (%)	Test (%)
90	83	97	95

The hardware and software results of CSFNN outputs are demonstrated in Table 7 for the random selected signature examples within Signature Dataset. The misclassified results were tabulated as bold characters. FPGA implementation and the pure software system gave similar classification performance for most random signatures. However some signatures are recognized in MATLAB correctly, some of them are misclassified in FPGA platform and vice versa.

Table 7. The results for random selected signature samples for signature dataset

Signature Samples	MATLAB Results			FPGA Results			Target Values		
	Out1	Out2	Out3	Out1	Out2	Out3	Out1	Out2	Out3
1	0.359	0.765	0.207	0.390	0.758	0.123	0.1	0.9	0.1
2	0.649	0.095	0.266	0.699	0.108	0.303	0.9	0.1	0.1
3	0.787	0.659	0.309	**0.868**	**0.548**	**0.500**	0.9	0.9	0.1
4	**0.665**	**0.510**	**0.527**	0.775	0.465	0.567	0.9	0.1	0.9

The CSFNN forward propagation delay was measured from RTL level circuit simulation. The response time computation starts when signatures are applied to CSFNN thus, the signature data transfer time was excluded. The computation rate performance for each platform is shown in Table 8. In order to compare performances, Giga Connections per Second (GCPS) was used as measurement metrics. CPS is defined as the number of operations that occur in the forward pass of a network per second [13]. It was seen from the Table 8, the pure software based system requires a higher computation delay for performing same task. Hardware systems are highly parallel, so they are able to execute several tasks at the same time. The intelligent system embedded on FPGA is approximately 10^5 times faster than its equivalent software implementation in terms of forward computation rates.

Table 8. CSFNN forward computational rates

Platforms	FPGA	Matlab
Forward Propagation Time (GCPS)	1.11	$10.4\text{x}10^{-6}$

5. DISCUSSION

In this study, intelligent algorithm based signature recognition system was implemented on FPGA. On-chip learning method is utilized in order to train CSFNN. DEA derivative free global optimization method is used to optimize CSFNN parameters during training and to avoid the problem of local minima in neural networks. CSFNN architectures were constituted on FPGA-based embedded high end Xilinx board using VHDL for signature databases. The weight, center and angle parameters of CSFNN were generated using training algorithm on FPGA environment and the updated parameters were transferred to CSFNN neurons. The utilization of the FPGA resource for the designed CSFNN – DEA architectures is less than the half capacity of the target platform. The high precision results are achieved by FPGA implementation using 16-bit FP IEEE 754-2008 standard. However, small quantization errors increase cumulatively during iterative training because of process workload. The main superiority of FPGA implementation is that embedded hardware accelerates the processing of CSFNN over software environment by inherent parallelism of FPGA. Other advantages of the proposed approach include rapid prototyping, flexibility in design modifications. When the commercial and the military demand a fast and a high accuracy signature recognition system, FPGA implementation of signature recognition system provides robust and satisfied performances.

Acknowledgement / Teşekkür

This study is supported by Yildiz Technical University Scientific Research Projects Coordination as 2012–04–03 KAP02 project number. The authors would like to thank Dr. Canan Cengiz from Kadir Has University is appreciated for providing the dataset.

REFERENCES / KAYNAKLAR

[1] Argones Rua E., Maiorana E., Alba Castro JL., Campisi P. (2012) Biometric Template Protection Using Universal Background Models: An Application to Online Signature. *IEEE Transactions on Information Forensics and Security;* 7, 269-282.

[2] Gruber C., Gruber T., Krinninger S., Sick B. (2010) Online Signature Verification With Support Vector Machines Based on LCSS Kernel Functions. *IEEE Transactions on Information Forensics and Security;* 40, 1088- 1100.

[3] Ketabdar H., Richiardi J., Drygajba A. (2005) Global feature selection for on-line signature verification. *In: International Graphonomics Society 2005 Conference;* 26-29 June 2005, pp. 59–63, Solerno, Italy.

[4] Erkmen B., Kahraman N., Vural RA., Yildirim T. (2010) Conic Section Function Neural Network Circuitry for Offline Signature Recognition. *IEEE Transactions on Neural Networks;* 21, 667- 672.

[5] Impedovo D., Giuseppe P. (2008) Automatic Signature Verification: The State of the Art. *IEEE Transactions on Systems, Man, and Cybernetics;* 38, 609-635.

[6] Al-Shoshan AI. (2006) Handwritten signature verification using image invariant and dynamic features. *In: Proceedings of the 2006 International Conference on Computer Graphics, Imaging and Visualisation;* 26-28 July 2006; pp. 173–176, Sydney, Australia.

[7] Armand S., Blumenstein M., Muthukkumarasamy V. (2006) Offline Signature verification based on the Modified Direction Feature. *In: Proceedings of the 18th International Conference on Pattern Recognition*; 20-24 August 2006, pp. 509–512, Hong Kong.

[8] Erkmen B., Kahraman N., Vural RA., Yildirim T. (2008) CSFNN Optimization of Signature Recognition Problem for a Special VLSI NN Chip. *In: Proceedings of the 3rd Int. Symposium on Communications, Control and Signal Processing*; 12-14 March 2008, pp. 1082-1085, St Julians, Malta.

[9] Yılmaz AR.., Erkmen B., Yavuz O. (2013) The Performance of Differential Evolution Algorithm for Training CSFNN Using a Pattern Recognition Application. *In: Proceedings of the 4th Int. Conference on Intelligent Control and Information Processing;* 9-11 June 2013; pp. 820-823, Beijing China.

[10] Merchant SG., Peterson GD. (2008) An evolvable artificial neural network platform for dynamic environments*. In: Proceedings of the 51st Midwest Symposium on Circuits and Systems;* 10-13 August 2008; pp. 77-80, Knoxville, Tennessee.

[11] Elitas M., Yavuz O., Erkmen B. (2012) Field Programmable Gate Array implementation of Conic Section Function Neural Network: An Alternative to Analog CSFNN Circuitry. *In: Proceedings of the IEEE 16th International Conference on Intelligent Engineering Systems*; 13-15 June 2012; pp. 135-138, Lisbon, Portugal.

[12] Shawash J., Selviah DR. (2013) Real-time non-linear parameter estimation using the Levenberg-Marquardt algorithm on Field Programmable Gate Arrays. *IEEE Transactions on Industrial Electronics*; 60, 170-176.

[13] Gomperts A., Ukil A., Zurfluh F. (2011) Development and Implementation of Parameterized FPGA-Based General Purpose Neural Networks for Online Applications. *IEEE Transactions on Industrial Informatics;* 7, 78-89.

[14] Savich AW., Moussa M., Areibi S. (2007) The Impact of Arithmetic Representation on Implementing MLP-BP on FPGAs: A Study. IEEE *Transactions on Neural Networks;* 18, 240 – 252.

[15] Yılmaz AR., Yavuz O., Erkmen B. (2014) FPGA Implementation of Differential Evaluation Algorithm for MLP Training. *In: Proceedings of the 2014 IEEE International Symposium on Innovations in Intelligent Systems and Applications*; 23-25 June 2014, pp. 425 – 430, Alberobello Italy.

[16] Sadykhov RK., Podenok LP., Samokhval VA., Uvarov AA. (2004) The system for handwritten symbol and signature recognition using FPGA computing. *Lecture Notes in Computer Science;* 3212, 447-454.

[17] Krid M., Dammak A., Masmodi DS. (2006) FPGA Implementation of Programmable Pulse Mode Neural Network with on Chip Learning for signature application*. In: Proceedings of the IEEE 13th International Conference on Electronics, Circuits and Systems;* 10-13 December 2006, pp. 942-945, Nice France.

[18] Senol C., Yildirim T. (2005) Signature Verification Using Conic Section Function Neural Network. *Lecture Notes in Computer Science;* 3733, 524 – 532.

[19] Dorffner G. (1994) Unified frameworks for MLP and RBFNs: Introducing Conic Section Function Networks. *Cybernetics and Systems: An International Journal* 25, 511-554.

[20] Erkmen B., Yildirim T. (2007) Obtaining Decision Boundaries of CSFNN Neurons using Current Mode Analog Circuitry. *In: Proceedings of the 18th IEEE European Conference on Circuit Theory and Design;* 27-30 August 2007; pp. 807-810, Seville Spain.

[21] Erkmen B., Vural RA., Kahraman N., Yildirim T. (2013) A Mixed Mode Neural Network Circuitry For Object Recognition Application. Circuits, Systems, and Signal Processing; 32, 29-46.

[22] Storn R., Price K. (1997) Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. The Journal of Global Optimization; 11, 341–359.

[23] Haykin S. Neural Networks. (1994) A Comprehensive Foundation. *New York, NY, USA: Macmillan College Publishing*.

[24] Rakvic RN., Ulis BJ, Broussard RP, Ives RW. (2009) Parallelizing Iris Recognition. IEEE Transactions on Information Forensics and Security 4, 812-823.

[25] Moreno F., Alarcon J., Salvador R., Riesgo T. (2009) Reconfigurable Hardware Architecture of a Shape Recognition System Based on Specialized Tiny Neural Networks With Online Training. IEEE Transactions on Industrial Electronics; 56, 3253-3263.