DESIGN OF FLOATING POINT ARITHMETIC UNITS AND ITS APPLICATION IN OFFLINE SIGNATURE RECOGNITION SYSTEM EMBEDDED ON FPGA

Nallathambi RAMYARANI [1], Veerana SUBBIAH [2]

[1]Sri Krishna College of Engineering and Technology, Coimbatore, [2]PSG College of Technology, Coimbatore, India.
[1]Kuniamuthur, Coimbatore, 9677559489, ramyarani@skcet.ac.in, [2]Peelamedu, Coimbatore, subbiah42@yahoo.com

Prabhakaran DEEPA [3]
Government College of Technology, Coimbatore, India.
Thadagam road, Coimbatore, deepap05@gmail.com

Abstract: *Floating point arithmetic circuits play an important role in scientific computing, signal and image processing applications due to its wide dynamic range and high precision. In this work, floating point Arithmetic and Logic Units (ALUs) architectures are designed, implemented on Field Programmable Gate Arrays (FPGAs) devices and utilized for signature recognition system. Synthesis results proved that log based unit provides faster computations, but increases the area compared with conventional floating point arithmetic units. Hence offline signature recognition system is designed using logarithmic single precision floating point arithmetic units and implemented on FPGA. The person's signature is classified by Support Vector Machine (SVM) and Neural Network (NN) approach. From the simulation results of the signature recognition system, the person's signature can be identified based on the features. The synthesis results proved that support vector machine classifier occupied only 34% of the FPGA resources available compared with a neural network approach that occupied 68% of the total resources. The various stages of the signature recognition system are analyzed in Xilinx FPGA and MATLAB.*

Keywords: *Field Programmable gate arrays, Floating point-arithmetic, Handwriting recognition, Support Vector Machines (SVM), Neural Network.*

1. Introduction

Floating point arithmetic and logic units are a part of the computer system. The requirements of floating point arithmetic have become very intense due to the dynamic range representation of real numbers and better precision compared to fixed point values. These floating point numbers are represented as per IEEE-754 standard 2008. This standard represents the basic single precision (32 bits), double precision formats (64 bits) and extended precision formats [1]. The complexity of these arithmetic circuits increases on hardware implementation due to increased bit width representation [2]. Logarithmic number systems (LNS) also provide a similar range and precision of floating point, but multiplication and division in LNS are modified to fixed-point addition and subtraction, respectively [3]. Nowadays, devices like Field Programmable Gate Arrays (FPGAs) are used for implementation of floating point arithmetic units because of their increased integration density and high performance operations. ASIC implementations produce high speed of computations and utilize less power. But it is very expensive to design, build and has very less flexibility after fabrication. On the other hand, FPGAs provide good speedup results and retain high flexibility after fabrication [4]. Hence the work is focused on the design and implementation of floating point arithmetic units on FPGA for embedding biometric system.

Signature recognition is one of the most popular research areas in personal identification and authentication. A person's identity can be verified in computer systems either based on the key, PAN card, ATM PIN number and password of the corresponding person. However, keys or cards may be stolen or lost easily. PIN numbers and passwords of the persons may be forgotten or disclosed to others. Hence, to achieve more reliable verification and identification, biometrics provides many methods of identity verification. In our society, the handwritten signature is considered as the primary means of identifying the signer of a written document. This method is one of the best ways to authorize transactions and verify the human identity compared with other electronic identification methods such as smart cards, RFID chips. The signature verification system depends on the selection of features and decision methodologies.

BUPT

There are nearly 40 different types of features used in signature verification. Features are classified as local and global. Global features are the features extracted from all the pixels pertaining to the signature image. Local features are extracted from a particular area of the signature image. Signature recognition and verification methods are classified into two type's namely online or dynamic verification systems and offline or static verification techniques.

In online Signature Recognition and Verification Systems (SRVS), features can be obtained by some special peripheral units like electronic tablet or personal digital assistant (PDA) for measuring hand speed and pressure on the human hand when the signature is created. On the other hand, an Off-line SRVS system depends on image processing and feature extraction methods. These static features are obtained either by camera or photo scanning of the signature [5]. Many classifiers like Hidden Markov model (HMM), Dynamic time warping (DTW), Support vector machines (SVM), Neural network (NN), Wavelet transform to Structural or syntactic methods are available for testing the image features. In this work, offline signature recognition system is designed using logarithmic floating point arithmetic units for classifying the person's signature using Support vector machines (SVM) and Neural Network (NN) approaches [6].

The paper is organized as follows: Section II describes the literature survey of floating point arithmetic units, signature recognition system, motivation and objectives of the work. Section III details the design of the log based floating point arithmetic units. Section IV presents the offline signature recognition system trained using SVM and neural network. Section V shows the experimental results and discussions. Finally, section VI presents the conclusion and future work.

2. Literature Survey

Many researchers proposed various methods for the design of floating point arithmetic units and offline signature recognition systems. Some of the works are presented here as a survey.

2.1 Floating Point and Logarithmic Arithmetic Designs

Ronald Scrofano et.al (2008) evaluated balanced and unbalanced binary tree arithmetic expressions using pipelined floating point cores. Implementation results on FPGA were compared in terms of area, speed and latency[7].Yee Jern Chong and Sri Parameswaran (2011) implemented multimode embedded floating point arithmetic units on FPGA. The embedded floating point units included the design of floating point adder and multiplier to perform double precision operations or two single precision operations simultaneously. Such designs provided performance and area benefits for the implementation of single precision and double precision floating point arithmetic units on FPGA [8].

Suganth Paul et.al (2008) proposed the method for computing log and antilog functions in FPGA device using Look-Up Table (LUT) along with interpolation. Logarithm function was computed using interpolation and thus the requirement of direct multiplication and division operations were avoided. Antilogarithm function was also computed using the LUT. These designs occupied less memory space compared with other works [9].Mark G.Arnold and Sylvain Collange (2011) presented an algorithm for Complex Logarithmic Number Systems (CLNS) that represented complex values in the log polar form. The real LNS hardware was used with CLNS along with the library parameters of floating point cores. Compared to CORDIC and Look-Up Table (LUT) approaches, CLNS provided better accuracy and efficient area[10].

2.2 Offline Signature Recognition Systems

Sharifah Mumtazah Syed Ahmad et al. proposed an automatic off-line signature verification system designed using statistical techniques. The Hidden Markov Model (HMM) technique was used to build a reference model for each local feature. The verification phase consisted of three layers of statistical techniques. FAR were computed as 22% and 37% for random and skilled forgeries respectively [11]. H. Baltzakis and N. Papamarkos, work was based on global, grid and texture features. Two stage Perceptron one-class one-network classification structure was implemented for each of the features. From the three feature sets, the classifier combined the decisive results of the neural networks and the Euclidean distance. These results were fed to a second-stage radial base function (RBF) neural network structure for the final decision. False Acceptance Rate (FAR) and False Rejection Rate (FRR) was found to be 9.81% and was 3% respectively [12].Abhay Bansal, Divye Garg, Anand Gupta presented the geometrical properties of the signature using contour matching

BUPT

algorithm[13]. Eight original signatures were trained and verified by the triangle matching algorithm. In Random Forgery, FAR was found to be 0.08% and 13.02% for Simple and Skilled forgery, 2.64% was computed for FRR.

Miguel A. Ferrer et al. [14] proposed an offline automatic signature verification system for geometric signature features. A 16 bit fixed-point arithmetic was used for feature set calculation and tested with different classifiers, such as Hidden Markov Models (HMM), Support Vector Machines (SVM) and Euclidean distance classifier. The experiments showed that using HMM, for random forgery FRR was found to be 2.2% and using SVM, FAR was found to be 2.65%. Using HMM, FRR and FAR was computed as 14.1% and 12.67% respectively. Hai Rong Lv et al. represented each of the signature images as landmark point set based on HMM. Grid features are extracted from grid partition technique. Pixels density and gravity center distance are used as some of the features for representing the grids of signature image [15].

M. Taylan Das and L. Canan Dulger proposed Particle Swarm Optimization (PSO) algorithm for neural network based off-line signature verification system. The three types of forgeries were used to test the performance of the algorithm. 40% of the signatures were detected correctly for skilled forgeries. Alan McCabe, Jarrod Trevathan and Wayne Read presented the methods for handwriting verification. Features like height, slant, and pressure are extracted and trained by the neural network. Several other approaches are compared with neural network for accuracy [16]. Ali Karouni, Bassam Daya and Samia Bahlak proposed the offline signature verification system. Geometric features like area, skewness and center of gravity were classified and verified by artificial neural network [17].

2.3 Motivation And Objectives Of The Work

Many research works proved that the implementation of conventional floating point multiplication and division modules on FPGA decreases the computing speed. On the other hand, logarithmic multiplication and division implementation on FPGA provides faster computations. However logarithmic addition and subtraction modules consume more area compared with conventional methods. The survey, based on offline signature recognition system proved that the design has not been implemented in real time embedded system and hence provides the possibility for the leakage of biometric information. Also in the survey works, the error rate was high in detecting the forged signatures. After defining the problem, the following objectives are set for the proposed work. I)To design arithmetic unit comprising of floating point addition, floating point subtraction, log based floating point multiplication and division with the objective of increasing the computing speed. II) To implement logarithm floating point unit based offline signature recognition system on FPGA for only the identification of the person's signature. The verification of the person's signature for detecting forgery signatures with a high accuracy rate can be implemented in MATLAB by creating Graphical User Interface. This part of the work is considered for future study by improving the feature extraction process in the designed system.

3. Log Based Floating Point Arithmetic Unit

Floating point arithmetic and logic unit consists of addition, subtraction, multiplication and division operations. In this work, the single precision format of IEEE 754 standard 2008 is used. Generally, floating point architectures are designed with the objective of optimization of area, speed and power [18, 19]. In Logarithm Number Systems (LNS), multiplication and division operations are reduced to addition and subtraction operations respectively. This property increases the speed of computations in logarithmic computations [20]. Hence log based floating point arithmetic units are designed in this work.

3.1 Log based Floating Point

Single precision log based floating arithmetic point is designed as shown in Fig.1. The design accepts two single precision inputs inp1 (X), inp2 (Y), clock, reset and select input decides the type of operation. The 1 bit f0, f1 indicates the flag for addition, subtraction module (f0=1) and multiplication division module (f1=1). Sel bit indicates the type of operation in addition (sel=0) subtraction (sel=1) module and multiplication (sel=0) division module (sel=1).

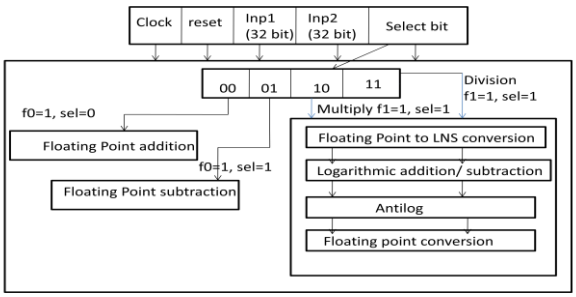


Fig.3.1. Logarithmic Arithmetic and Logic Unit

BUPT

3.2. Addition / Subtraction

Floating point addition and subtraction operations are evaluated as follows:

Let the two operands be X and Y in IEEE 754 format (32 bit).

1. To get the sign bit (1 bit)

If two operands have similar sign bits, resultant sign bit will be the same i.e. 0 or 1.

If the sign bit differs, exponent and mantissa fields are compared. If the first operand is larger than the second, the first operand sign bit will be the resultant and vice versa. If both the operands have equal exponent and mantissa fields, the resultant sign bit is 0.

2. Exponent comparison (8 bits)

The exponent (e) difference of X and Y is computed. The 24^{th} bit (hidden) is made explicit in the mantissa.

If Xe >Ye, the mantissa of Y is right shifted as per the exponent difference value and its exponent value is incremented. The left most digits are filled with zeros.

If Ye > Xe, the mantissa of X is right shifted as per the exponent difference value and its exponent is incremented. The left most digits are filled with zeros.

3. The aligned mantissas are either added or subtracted depending upon the type of operation.

4. Exceptions overflow and underflow are analyzed with the resultant mantissa values. In case of overflow, the computed mantissas are shifted to the right once and 1 is added to the exponent. In case of underflow, the computed mantissa is shifted to the left until the first binary 1 is detected. The number of left shifts is subtracted from the exponents.

3.3. Multiplication / Division

Multiplication and division in LNS become addition and subtraction respectively, due to the logarithmic property given below:

$$\log_2(X \times(\div) Y) = \log_2(X) \pm \log_2(Y) \quad (1)$$

Logarithm of the two numbers $\log_2(X)$ and $\log_2(Y)$ are computed as per the following conversions and the resultant logarithm values are either added or subtracted depending upon multiplication and division operations respectively. Antilog of the resultant value is calculated. The final result is converted back to floating point format using fixed to floating point converter. By *XORing* between *X& Y, the sign bit is calculated.*

3.4. Floating point to LNS conversion

The conversion from floating point to LNS involves two steps that can be done in parallel. The floating point input contains three parts, namely Sign (S), Exponent (E), Mantissa (M) and output LNS gives two parts namely Exponent (E) and Mantissa (M).

The floating point number 'X' is defined as

$$X=S\times2^{E}\times M \quad (2)$$

The logarithm of a number is

$$\log_2(X) = \log_2(2^{E} \times M) \quad (3)$$

By multiplication rule of log;

$$\log_2(X) = \log_2(2^{E}) + \log_2(M) \quad (4)$$

E value is considered without bias

By power rule of log;

$$\log_2(X) = E \times \log_2(2) + \log_2(M) \quad (5)$$

$$\log_2(2) = 1;\ \log_2(X) = E + \log_2(M) \quad (6)$$

Mantissa is considered with hidden bit `1' i.e. 1.M. The log of number is represented as

$$\log_2(X) = E + \log_2(M) \quad (7)$$

The logarithm of the mantissa values are obtained from the pre-computed values stored in LUT and added with the exponent [21]. The Look-Up Tables (LUTs) values are generated from MATLAB and the values are stored in block RAM memory of FPGA board. The size of the lookup table is dependent on the number of bits of accuracy considered after fraction point (mantissa bits). For example, for n bit of accuracy, LUT depth required is 2^n. For example, if 12 bits of accuracy after the fraction point is required, then the LUT depth needed will be 0 – 4095 and each location will contain 32 bits of data. The LUT is addressed by n bits of mantissa. So the first n bits of the mantissa are given as an address to the LUT that contains the required data and the LUT output will be the corresponding log value for the mantissa. Once the logarithm addition or subtraction of the numbers is calculated depending on the type of operation, the antilog of the value is computed. The anti-log is the inverse log calculation.

The antilog of a log number is calculated as

$$\text{Antilog}_2(E+\log_2(M))=2^{(E+M)} \quad (8)$$

$$\text{Antilog}_2(E+\log_2(M))=2^{(E)}\times2^{(M)} \quad (9)$$

The antilog values of mantissa values are obtained from the pre computed values stored in the LUT and multiplied with 2^E. This multiplication is performed by shifting the number obtained from LUT by the exponent value. The resultant data is converted back to floating point format by finding the first 1 from the Most Significant Bit (MSB) side. Accordingly

BUPT

the data is shifted and the exponent value is obtained either by adding or subtracting 127 to the shifted number. The rest of the bits represent the mantissa part, except the first 1 from MSB. Addition or subtraction of 127 to the shifted number depends on the direction of shifting of negative and non negative values.

The same procedure is followed in the design of double precision log based floating point arithmetic unit with the change in bit width. Double precision standard consists of 64 bits, namely 1 sign bit, 11 bit exponent and 52 bit mantissa. A Multimode architecture is developed in this work by integrating both single precision and double precision log based floating point arithmetic unit. This architecture can be executed either in single precision or with double precision or in both the modes simultaneously depending on the selection bit. By simultaneous execution of both the modes, this architecture can reduce the combinational path delay of executing the designs separately.

4. Offline Signature Recognition System

This section describes the stages involved in an offline signature recognition system. The system is composed of various stages like data acquisition, preprocessing, feature extraction, training and identification as shown in Fig.2. [26]. Preprocessing and feature extraction stage are performed in Xilinx FPGA. The features are trained using Support Vector Machine (SVM) and Neural Network (NN) in MATLAB. Classification stage for the identification of the person's signature is performed in Xilinx FPGA. The mathematical computations involved in the classification stage are manipulated using log based single precision floating point arithmetic units for support vector machine and neural network approaches**.**

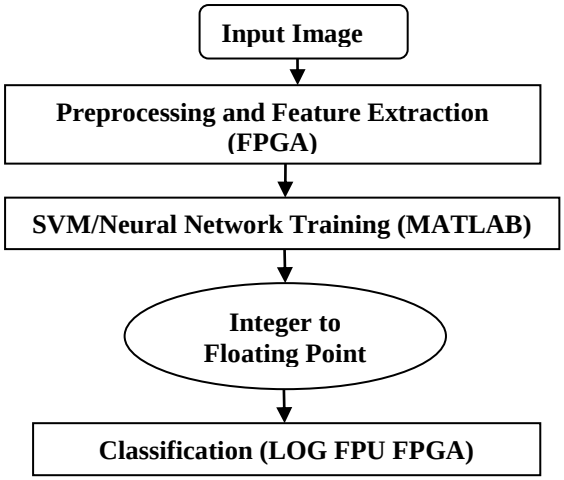


Fig.4.1. Flow Diagram

4.1. Data acquisition

In this work, signature of two persons Abhit 'a' and Gaurav 'g' each in 10 different styles are handwritten on a paper and the scanned image is obtained in the size of 240×150 as shown in Fig.3. The 20 images are converted into coefficient (.coe) files as Xilinx FPGA can't access the image files directly. In MATLAB, code is written to read the input image and resized. A new file is created to open image files in .coe format. The pixel values are obtained from 'for' loop. The data's are entered in binary format. Once the coe file is created in MATLAB, these .coe files are stored in the block RAM memory of FPGA using IP core provided in Xilinx design suite.

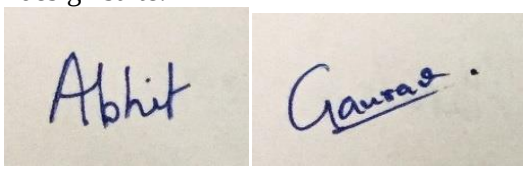


Fig.4.2. Input image 'a' Input image 'g'

4.2 Preprocessing and Feature Extraction in FPGA

The preprocessing and feature extraction stages are performed in Xilinx FPGA. The accuracy of features extracted from an image is improved in the signature pre-processing stage. In this stage the images are binarized to remove noise and thinned to remove the thickest of differences of the pen. Images are made to have one pixel thick after pre-processing stage as shown in Fig.4.

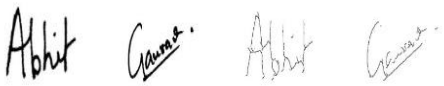


Fig.4.3 Binary Images Thinned Images.

In order to extract some of the features the images are required to be cropped. Hence, the 20 input images of size 240×150 are cropped so that the image covers the area that contains only the signatures as shown in Fig.5.

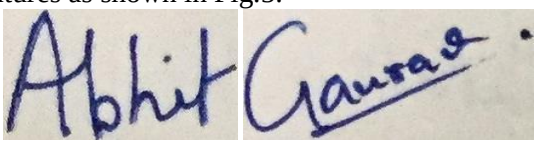


Fig.4.4. Cropped Image 'a' Cropped Image 'g'

Global features like height, width, centroids of 4 elements, quadrant areas of 4 elements, Gray Level Co-occurrence Matrix (GLCM) of 4 elements, edge point number, horizontal histogram, vertical histogram and Histogram of oriented Gradients (HOG) are extracted from the image. HOG containing 81 elements of an image is obtained from

BUPT

the cropped and resized (binary and thinned) image, whereas the remaining mentioned global features containing 17 elements are obtained from the cropped image. The feature vector size represents the addition of 81 elements of HOG and 5 times the global features of 17 elements. Thus the total of 81 elements of HOG and 85 elements of global features constitute the total size of the feature set as 1×166. These features are extracted as a text file using Xilinx design suite in Verilog. As 20 images are considered in this work, the size of the feature mix matrix used for training is 20×166.

4.3 Training Using SVM in MATLAB

In this stage, feature vector obtained in the previous stage is trained using support vector machine for classification in the next stage. SVM is a machine learning algorithm that maps the feature vector to a higher dimensional plane by non-linear mapping. It is a binary classifier that determines the linear hyper plane for the classification of two classes.

The feature mix matrix of size 20×166 is used for training the images using support vector machine. The output 20×2 matrix groups the 20 input images into two classes [27]. The values assigned are '01' and '10' for the two classes. The kernel function svm train maps the training data to the kernel space. SVM struct file is the resultant output obtained after training the features. This file consists of support vectors, bias, alpha, kernel function, shift and scale factors used for feature normalization. These data's are converted to IEEE 754 32 bit floating point standard.

4.4.Classification in Xilinx FPGA

This stage is performed in Xilinx FPGA for the recognition of a person's signature. Support vector machine provides two classes for the identification of the person 'a' and 'g'. The support vectors of size 7x166, alpha of size 7x1, bias, shift 1x166, scale 1x166 and sample test inputs (1x166) are given as input in IEEE 754 32-bit floating-point format.

The equation used in SVM classify is

$$c = \sum_i \alpha_i k(s_i, x) + b \qquad (10)$$

Where s_i is the support vector, α_i is the weight, b is the bias, and k is a kernel function that represents the dot product. The sample features are normalized using shifting and scaling operations. Shifting operation represents the row vector containing the negative of the mean across all observations in training. Scale factor represents the row vector containing the inverse of standard deviation of the training. After normalization, dot product of support vectors and normalized feature is calculated. The final value of SVM is manipulated by multiplying the alpha value with SVM dot product of size 1×7. Then the bias value is added to it. These computations involving addition and multiplication operations are manipulated using the designed log based single precision floating point arithmetic unit. The final value of SVM obtained if greater than or equal to zero, the index value is '10' that represents the second person's signature ('g'). If the resultant SVM value is less than zero, the index value is '01' that identifies the first person's signature ('a').

4.5. Training Using Neural Network in MATLAB

In this section, the feature mix matrix obtained in the preprocessing and feature extraction stage is trained using neural networks [28]. Neural networks are widely used for pattern recognition, object classification, medical diagnosis, etc. Multilayer feed forward network is used in this design. A multilayer network consists of one input layer, one or more hidden layers and one output layer. In a feed forward network, the signal moves in only one direction between the neurons of each layer to other layers.

Supervised learning method is used in this work for distinguishing the person's signature. i.e. in addition to the input features, the network has to know the output vector size required for training. A feature mix matrix of size 1×166 is given as input. The output 20×2 matrix is used to group the 20 images into '2'. The values are taken as '01' and '10'. The feed forward neural network with a single hidden layer used for training is shown in Fig.6. The hidden layer is trained based on sigmoid activation function.

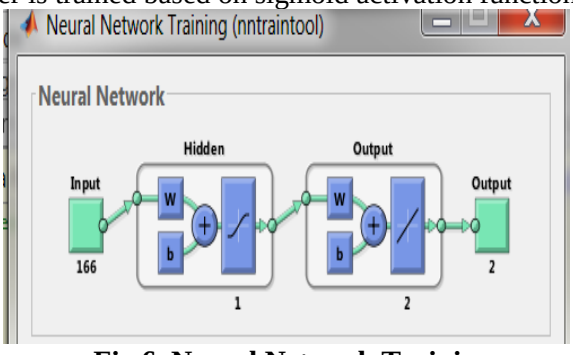


Fig.6. Neural Network Training

The input bias, output bias, input weight and output weight are obtained after training in MATLAB. These parameters are utilized for classification in the testing part to determine the hidden sigmoid function and output values.

BUPT

4.6. Testing Using Neural Network in Xilinx FPGA

The classification stage for the identification of the person is performed in Xilinx FPGA. The computations of normalized value, hidden layer, sigmoid function and output layer are manipulated using log based single precision floating point arithmetic unit. The output value matched with the correct signature feature of a person will be greater than the other output value. The minimum and maximum values of the features trained in the previous stage are normalized. When the maximum and minimum values are equal, the normalized value is

$$norm_x = y_range \times (feat - Xmin) \quad (11)$$

Where feat= feature, Xmax and Xmin= maximum and minimum value, output y_range = 2, Ymin = -1. When the maximum and minimum values are not equal,

$$normalized_x = norm_x / (Xmax - Xmin) + Ymin \quad (12)$$

The input bias, output bias, input weight, output weight, minimum, maximum values of size 1×166 and sample test inputs of size 1×166 are converted to IEEE 754 32 bit width floating point and given as input in this stage. The normalized value of feature size 1×166 is used to calculate the single hidden layer [29].

$$\text{Hidden layer} = (\text{Input weight} \times \text{normalized value}) + \text{Input bias.} \quad (13)$$

The hidden layer of size 1×1 is obtained. The tangent sigmoid function is determined using the formula,

$$\text{Sigmoid function} = \frac{\text{Hidden layer}}{1 + \frac{(\text{Hidden layer})^2}{3 + \frac{(\text{Hidden layer})^2}{5 + \ldots}}} \quad (14)$$

Finally the output layer is calculated using the relation,

$$\text{Output layer} = (\text{output weight} \times \text{sigmoid function}) + \text{output bias.} \quad (15)$$

The output values of the two layers are compared. The second person ‘g’ signature is determined if the second output layer is greater than the first output layer and vice versa case represents the first person's signature ‘a’.

5. Experimental results

Log based floating point arithmetic units was designed and the performance analysis in terms of area and speed was compared with the conventional floating point designs. These modules were implemented on Xilinx Virtex 6 FPGA. Fig.7.represents the simulation results of log based floating point arithmetic unit. Designs were simulated in single precision mode. All the four arithmetic operations were computed.

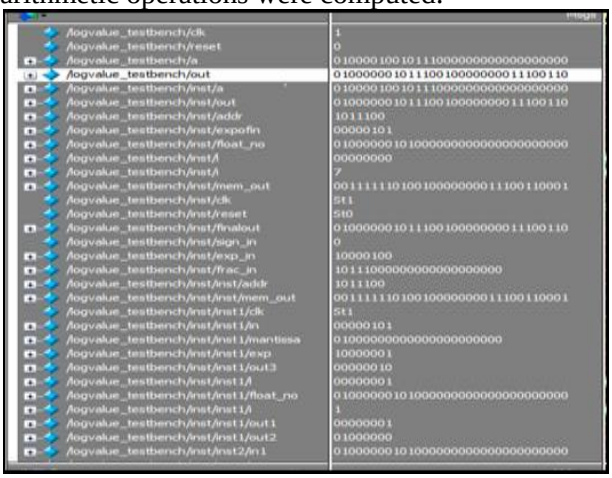


Fig.5.1 Log based Floating Point Arithmetic Unit

Table 1: Comparison among floating point Arithmetic Unit Designs on FPGA

Synthesis Parameters (VIRTEX 6 FPGA)	Conventional FPU	Log based FPU
Slice Registers	239 out of 93120	683 out of 93120
LUT-FF pairs (slices)	2123 out of 46560	1484 out of 46560
Maximum Frequency (MHZ)	22.24	219.296
Latency (ns)	21.67	8.32

Table 1 proved that log based computations performs faster on VIRTEX 6 FPGA as floating point multiplication and division was reduced to logarithmic floating point addition and subtraction respectively. The proposed design increased the speed of computations to 219.296 MHz, but the area was increased slightly higher when compared with conventional floating point arithmetic units. This is due to the fact that, the optimization of speed and area parameters is a tradeoff in VLSI designs. Latency that represents the time interval between the

BUPT

input and output generated is also decreased in the proposed design due to this property.

5.1 Simulation Results Using SVM for Signature Recognition

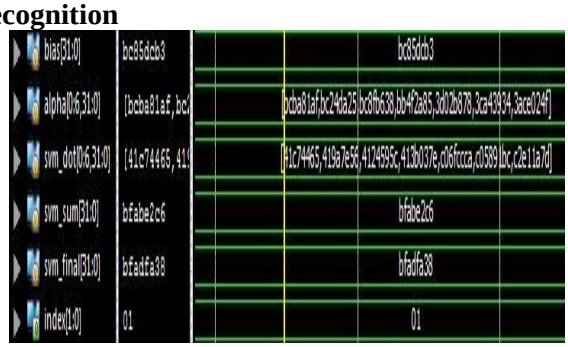


Fig.5.3. Identification of person 'a1' signature.

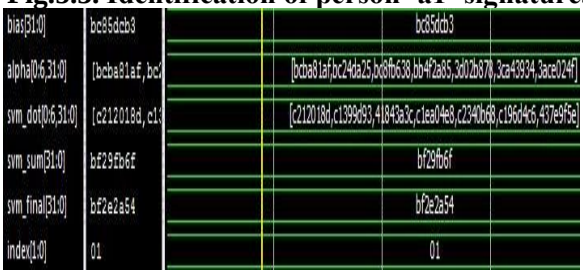


Fig.5.4. Identification of person 'a11' signature

Fig.9. and Fig.10. represents the simulation result in the identification of the person 'a'. The value of SVM final is negative that results in the index value '01'. 'a11'represents the cropped signature of person 'a'. The SVM final value can be matched with the block memory value to verify the identity of the person with feature input.

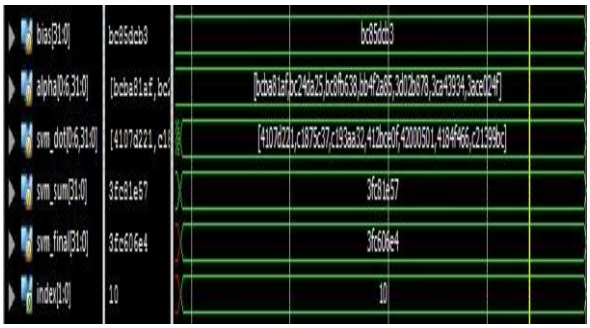


Fig.5.5. Identification of person 'g' signature

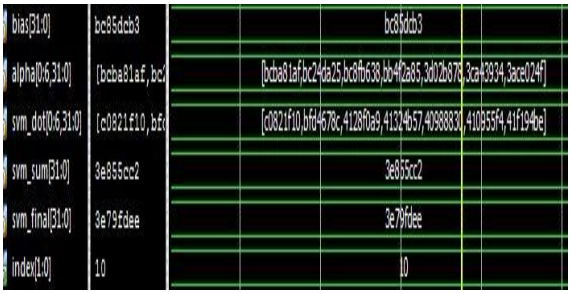


Fig.5.6.Identification of person 'gf6' signature

Fig.11. and Fig.12. represents the simulation result in the identification of the person 'g'. The value of SVM final is positive that results in the index value '10.'gf6'represents the cropped signature of person 'g'. The SVM final value can be matched with the block memory value to verify the identity of the person with feature input.

5.2 Simulation Results Using Neural Network for Signature Recognition

Fig.13. indicates the simulation result in the identification of the person using neural network. The index value indicates '01' that determines the first person's signature 'a'.

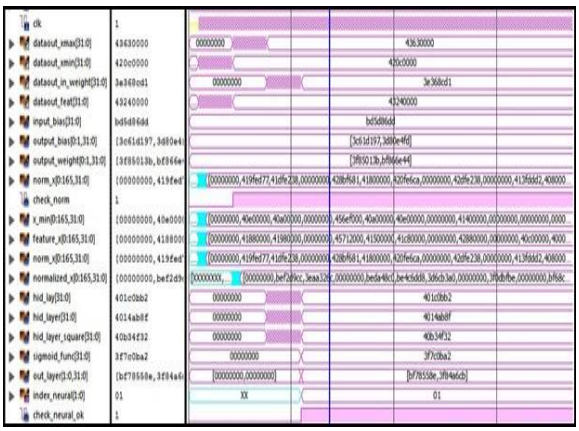


Fig.5.7 Identification of Person 'a'.

The value '01' was assigned to the feed forward neural network during training stage. Fig.14. indicates the simulation result in the identification of the second person signature using neural network. The index value indicates '10' that determines the second person signature 'g'. The value '10' was assigned to the feed forward neural network during training stage.

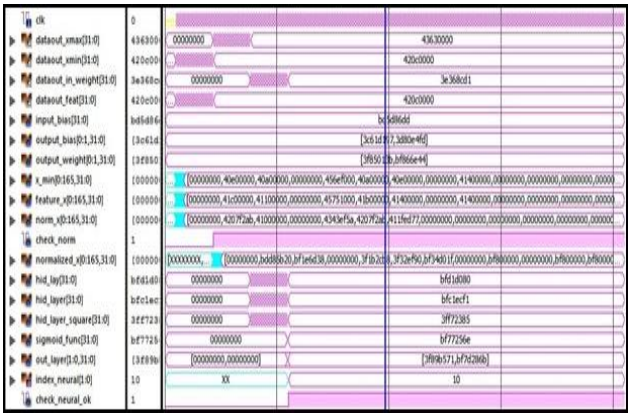


Fig.5.8. Identification of Person 'g'.

BUPT

Table 2: Comparison between SVM and Neural Network Classifier

Synthesis Parameters (VIRTEX6 FPGA)	SVM	NEURAL NETWORK
Slice Registers	133	252
Slice LUTs	23433	38802
Bonded IOBs	3	3
Block RAM	8	2
Maximum Frequency (MHz)	23.808	14.400

Table 2 provided the comparison for the testing stage of support vector machine and neural network classifiers for signature recognition system embedded on FPGA. Results showed that support vector machine classifier occupied less number of hardware resources and provided faster computations compared to neural network implementation.

6. Conclusion

This work described the design of floating point arithmetic units for offline signature recognition application. Log based floating point ALU's and multi mode log based arithmetic architectures were designed and implemented on Virtex FPGA. The comparison results showed that log based computations was performed faster, but occupied more hardware resources than other designs. In the field of image processing, the log based single precision floating point modules were utilized to design offline signature recognition system embedded on FPGA. The system was analyzed based on 4 stages. In the preprocessing stage, images were binarized and thinned. Features were extracted and a feature mix matrix of size 20×166 was given as input for SVM and Neural Network training in MATLAB. From the simulation results of classification stage in Xilinx FPGA, the person's authentication was identified. The FPGA synthesis results of signature recognition system using both the approaches proved that Support Vector Machine classifier occupied less number of hardware resources compared with neural network implementation.

In future, the work will be extended to train the designed signature recognition system using other classifiers. Signature forgeries can be determined accurately by improving the performance of the feature extraction process. Different types of forgeries involved in the signature recognition system can be identified. The modeled floating point arithmetic units can be applied in the field of digital signal processing.

References

1. IEEE Standard for Binary Floating-Point Arithmetic, *IEEE Standards Board, IEEE*, New York, 2008, Tech. Rep. ANSI/IEEE Std. 754, 2008, pp 1-58.

2. Jonathan Ying Fai T., David N., Rob Rutenbar, A.: '*Reducing power by optimizing the necessary precision/range of floating-point arithmetic*', IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.28,no.3, June 2000, pp.273-286.

3. Chen, C., Chen, R. and Yang, C.: '*Pipelined computation of very large word length LNS addition/subtraction with polynomial hardware cost'*, IEEE Transactions on Computers, vol.49, no.7, July 2000, pp. 716–726.

4. Chi Wai Y., AlastairSmith M., Wayne L., PhilipLeong, H.W., and Steven Wilton,J.E.: '*Optimizing floating point units in hybrid FPGAs'*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.20, no.7,July 2012, pp. 1295-1303.

5. Plamondon, R., and Srihari, S.N.: '*Online and offline handwriting recognition: A comprehensive survey*', IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no 1, Jan. 2000, pp. 63-84.

6. Müller K.R., Mika S., Rätsch G., Tsuda K., and Schölkopf B.: '*An introduction to kernel-based learning algorithms*', IEEE Trans. Neural Networks, vol. 12, no. 2,March 2001, pp. 181-201.

7. Scrofano R., Zhuo L., Prasanna V.K.: '*Area efficient arithmetic expression evaluation using deeply pipelined floating point cores'*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.16, no.2, Feb.2008, pp.167-176.

BUPT

8. Chong Y.J., Parameswaran S.: '*Configurable multimode embedded floating-point units for FPGAs'*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.19, no.11, Nov.2011, pp.2033-2044.

9. Paul S., Jayakumar N and khatri S.P.: '*A fast hardware approach for approximate, efficient logarithm and antilogarithm computations*', IEEE Transactions on Very Large Scale Integration (VLSI) Systems,vol.17, no.2,Feb.2009, pp.269-277.

10. Arnold M., Sylvain C.: '*A real/complex logarithmic number system ALU'*, IEEE Transactions on Computers, vol.60, no.2, Feb.2011, pp.202-213.

11. Ahmad S.M.S., Shakil A., Faudzi M.A.: '*A hybrid Statistical Modeling, Normalization and Inferencing Techniques of an Off-line Signature Verification System'*, World Congress on Computer Science and Information Engineering, 31st March-2 April 2009, pp.491-501.

12. Baltzakis H., and Papamarkos N.: '*A new signature verification technique based on a two-staged neural network classifier'*, Engineering Applications of Artificial Intelligence, June 2001, pp. 95-103.

13. Bansal A., Garg D., and Gupta A.: '*Pattern matching classifier for offline signature verification*', First International Conference on Emerging Trends in Engineering and Technology (IEEE Computer Society), 16-18 July 2008, pp.250-256.

14. Miguel Ferrer A., Jesu´s Alonso B., and Carlos Travieso M.: '*Offline geometric parameters for automatic signature verification using fixed-point arithmetic'*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 6, June 2005, pp. 993-997.

15. Rong Lv H., Jun Yin W., and Dong J: '*Offline signature verification based on deformable grid partition and hidden markov models'*, IEEE International Conference on Multimedia and Expo ICME 2009, New York, 28th June-3rd July 2009, pp.105-110.

16. McCabe A., Trevathan J., and Read W: '*Neural network-based handwritten signature verification'*, Journal of Computers, vol. 3, no. 8, August 2008, pp9-21.

17. Karouni A., Daya B., Bahlak S: '*Offline signature recognition using the Neural Network's approach*', Procedia Computer Science, Elseiver, vol.3,2011, pp. 155–161.

18. Galal S., and Horowitz M: '*Energy-efficient floating-point unit design*', IEEE Transactions on Computers, vol. 60, no.7, July 2011, pp. 913-922.

19. Lee D.U., Gaffar A.A., Cheung, R.C.C.: '*Accuracy guaranteed bit-width optimization'*, IEEE Transactions on Computer Aided Design, Oct 2006, pp. 1990-2000.

20. Alachiotis N., and Stamatakis A: '*Efficient floating-point logarithm unit for FPGAs'*, IEEE International Symposium on Parallel & Distributed Processing**,** Atlanta, USA, 19-23 April 2010, pp.978-984.

21. Fu H., Mencer O., and Luk W: '*FPGA designs with optimized logarithmic arithmetic*', IEEE Transactions on Computers, vol. 59, no. 7, July 2010, pp. 1000-1006.

22. Yee Jern Chong, and Sri Parameswaran : 'Custom Floating-Point Unit Generation for Embedded Systems', IEEE Transactions On Computer-Aided Design Of Integrated Circuits And Systems, vol. 28, no. 5, May 2009, pp. 638-650.