Asian Journal of Information Technology 7 (5): 191-200, 2008
ISSN: 1682-3915

Pulse Mode Neural Network Implementation for Handwritten Digit Recognition

Alima Damak Masmoudi, Mohamed Krid and Dorra Sellami Masmoudi
Computers Imaging Electronics and Systems Group (CIELS), ICOS Research Unit,
University of Sfax, Sfax Engineering School, BP W, 3038 Sfax, Tunisia

Abstract: This study describes a new pulse mode artificial neural network (PNN) implementation based on floating point number format. For on-chip learning operations, the back-propagation algorithm is modified to have pulse mode operations for effective hardware implementation. By using floating point number system for synapse weight value representation, any function can be approximated by the network. The convergence rate of the learning and generalization capability is improved. The proposed network is applied for digit recognition application. The recognition approach is based on a series of features, which are at most independent of orientation and position. The most important featurea are based on Zernike moments. However, an exclusive use of Zernike moments in digit recognition increases tremendously the neural network size, since higher orders are needed to ensure best recognition rates. Moreover, given their geometrical invariance, Zernike moments give the same description to some different digits such as 6 and 9. Thus, we make use of other features based on structural descriptors witch are the terminating point number and the terminating location number which is orientation dependent. This features based presentation of the digits reduces the required Zernike order and the number of hidden layers and adds a great simplicity to the design, making possible the on-chip learning implementation for online operations. The proposed PNN is implemented on a Virtex II FPGA platform. Various experiments are carried on for design evaluation.

Key words: Pulse mode, synapse multiplier, floating point operation, handwritten digits, zernike moments, endpoint detection, FPGA implementation

INTRODUCTION

Although most applications of neural networks are carried out using software simulators, many other potential applications require large, high-speed networks implemented in efficient custom hardware, which can fully use the inherent parallelism embedded in neural network dynamics.

Spiking neurons depart from traditional connectionist models in the sense that the information is transmitted by the means of pulses (or spikes), rather than continuous mode neurons. This may allow spiking neurons to have richer dynamics and to exploit the temporal domain to encode or retrieve information in the exchanged spikes. Hardware implementations of spiking networks may bring benefits such as very low power consumption.

One of the effective approaches to carry out the neurological network material execution is the architecture based on pulse mode operations (Reyneri, 1995). Many of these architectures present an elegant and compact solution (Marchesi and Orlandi, 1993; Lehmann *et al*., 1993) over early continuous mode solutions (Krid *et al*., 2005). By coding data in the frequency domain, pulse mode operations are based on simple frequency multipliers rather than the conventional overshooting multipliers, used in continuous mode neural networks. Such compact solution offers an important feature to pulse mode neural networks with respect to on-chip learning applications. Owing to their advantages, they acquire nowadays a growing importance in neural network implementation (Maeda and Tada, 2003; Martincigh and Abramo, 2005).

A pulse mode digital architecture has been proposed in Moon *et al.* (1992) and Ikenaga and Ogura (1998). Taking advantage of the compactness of the solution proposed by Hikawa (1999a), in a multiplierless architecture is applyin (Damak *et al*., 2006; Krid *et al*., 2006) which the synapse is made up witha Direct Digital Frequency Synthesizer (DDFS) and the neuron uses a nonlinear adder. The limitation of this approach is that the weight range of the synapse multiplier is limited between -1 and 1, which makes learning difficult, especially in networks with high input resolution. Indeed, given one neural network architecture, universal approximation

Corresponding Author: Alima Damak Masmoudi, Computers Imaging ELectronics and Systems Group(CIELS), ICOS Research Unit, University of Sfax, Sfax Engineering School, BP W, 3038 Sfax, Tunisia

features of neural networks states that there exists a number of hidden layers, for witch weights can be adjusted to meet any input-output relation. However, there is no rule specifying the weights range. Generally, high weight values arise from high data resolution.

To a certain extent, this weight range limitation was solved by using neurons with adjustable nonlinear activation function (Hikawa, 1999b). Steeper nonlinear activation function has the same effect as the larger weight value has. However, some neural network applications still require much larger synaptic weights. In Hikawa (2000), the pulse mode neural network based on floating point operation is presented. Owing to pulse mode operations, a simple programmable frequency converter can be used instead of the multiplier. Pulse signal is used to update the weight values. Weight value in the floating point format is stored in a special up-down counter so that the synaptic weight can be updated easily.

In this study, the proposed system uses simple floating point number to represent the synaptic weight values. An improved synapse unit is proposed without any high hardware cost. Using floating point operations, synaptic weights cover a very wide range thus providing PNN applications with high precision.

Handwritten digit recognition has been an active area of research and development for at least two decades. The handwriting is one of the most familiar communication media. Pen based interface combined with automatic handwriting recognition offers a very easy and natural input method (Oulhadj *et al*., 1999).

In this study, we applied the proposed PNN to a handwritten digit recognition learning application. This approach is based on Zernike moments (Shutler and Nixon, 2006). For best recognition, high order moments are required, which increases tremendously the neural network size, stating the use of more hardware resources. Moreover, as with regular moments, some forms of invariance in Zernike moments may be a disadvantage in recognition since, for instance, digits "6" and "9" are distinguishable only by direction. Thus, for those reasons, we were seeking other features, for the use of more compact implementation. In our model, network inputs are accordingly the different features and the output is a binary vector describing the corresponding digit. The model is constructed by using a two-layer neural network. For training and generalization test, a handwritten digit data base of eighty samples was used. The whole system is implemented on a virtex II field programmable gate array (FPGA) platform and the neuron characteristics are tested experimentally in Fig. 1.

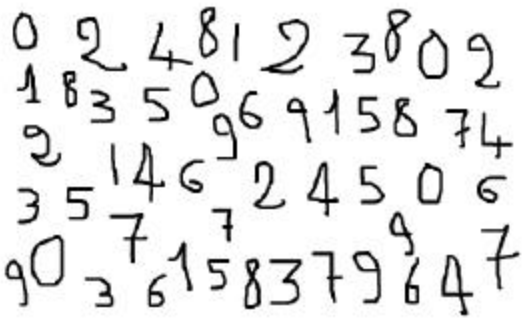

Fig. 1: Some data base samples

OVERVIEW OF THE DIFFERENT DIGIT RECOGNITION FEATURES

The objective of handwritten digit recognition is to develop a system which approaches the human capacity in reading. The recognition took a real take-off while being implied in several technological sectors. In this study, we present an improved method for handwritten digit recognition using Zenike moments and other discrete structural descriptors such as the terminating point number and the terminating location umber.

Implementation of zernike moments: Implementation step of Zernike moments are illustrated in Fig. 2.

Binarisation: Binary images are typically obtained by thresholding a grey level image. Binary images are quantized to two values, usually denoted by 0 and 1, but often with pixel values 0 and 255, representing black and white shown in Fig. 3.

Segmentation: The registered and preprocessed input data has to be subdivided into subparts to create meaningful entities for classification. This stage of processing is called segmentation. It may either be a clearly separate process or tightly coupled with the previous or following processes. In either case, after the pattern recognition system has completed the processing of a block of data, the resulting segmentation of the data can be revealed.

We have segmented our data base by subdiving into different connexe regions. For each region, a set of properties are computed such as centrod, area, minimal bounding box, etc... Then based on these properties, each connexe region is represented separately.

Redimensionning: The size of the data is subject to a normalization of height and width. The normalization is based on the principles of under-sampling or over-sampling shown in Fig. 4.

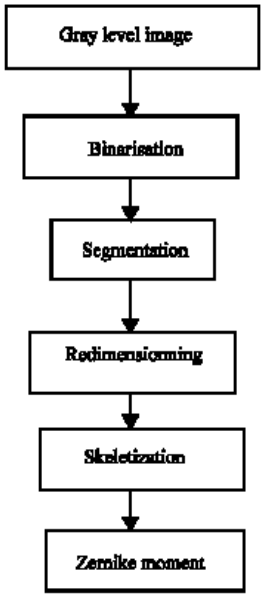

Fig. 2: Implementation of Zernike moments

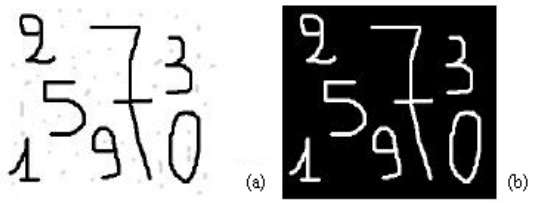

Fig. 3: Example of an image (a) and its binarisation (b)

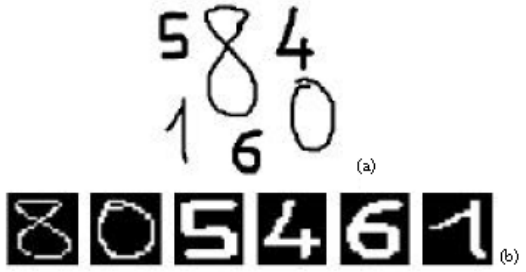

Fig. 4: Example of an image (a) and its segmentated and redimentionned included digits (b)

Skeletonization: A skeleton of an image can be used as a starting point for feature extraction. The skeletonisation process, also known as thinning or Medial Axis Transform is by itself problematic regarding both the computational aspects and the potential uniqueness of the result. Figure 5 shows a handwritten digit of image example and its skeleton.

Fig. 5: Example of a digits image before and after skeletonization

In the context of handwritten digit recognition, the goal of the thinning process is quite obvious: the trace of a pen spreading ink while moving on a paper should be recovered from the final image. Moreover, thinning is a kind of normalization, aiming at increasing Zeninke moment sensitivities for better recognition. The average line width can be estimated prior to thinning and that information is used in skeletonisation. Alternatively, the average line width can be estimated after thinning if the information is needed in further processing. Vectorization is a step that commonly follows skeletonisation. In vectorization, the pen trace is represented with a small number of linear line segments known as strokes.

Zernike moments: Zernike moments are based on a set of complex polynomials that form a complete orthogonal set over the interior of the unit circle (Khotanzad and Hong, 1990). Zernike moments are defined to be the projection of the image function on these orthogonal basis functions. Complex Zernike moments are constructed using a set of complex polynomials which form a complete orthogonal basis set defined on the unit disc $(x^2 + y^2) \leq 1$. The Complex Zernike moments are defined as:

$$A_{mn} = \frac{m+1}{n} + \sum\sum f_{xy}\, V_{mn}(x,y) \qquad (1)$$

where, $m = 0, 1, ..., \infty$ defines the order. While, n is an integer (that can be positive or negative) depicting the angular dependence, or rotation, subject to the conditions

$$m - |n| = \text{even}, \; |n| \leq m \qquad (2)$$

The Zernike polynomials $V_{mn}(x, y)$ expressed in polar coordinates are

$$V_{mn}(r, \theta) = R_{mn}(r)\, e^{jn\theta} \qquad (3)$$

where, (r, θ) are defined over the unit disc, $j = \sqrt{-1}$ and $R_{mn}(r)$ is the orthogonal radial polynomial, defined as:

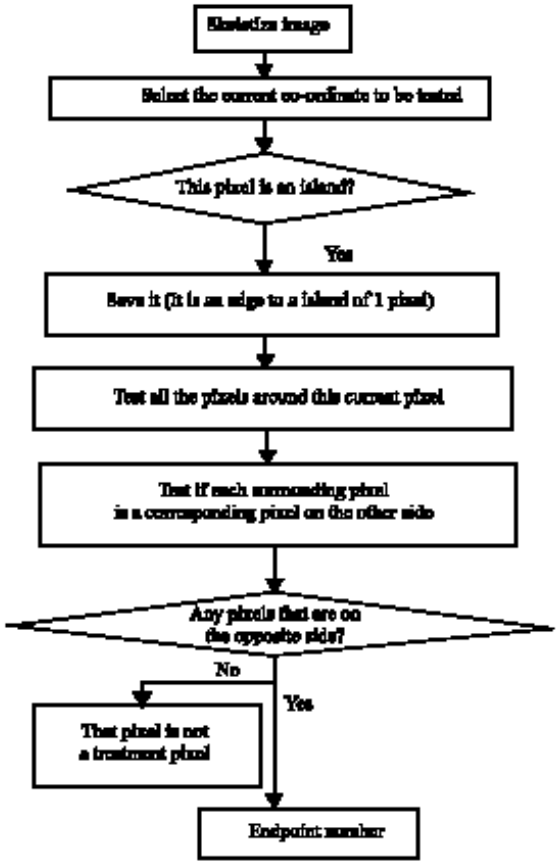

Fig. 6: Step of endpoint evaluation

$$R^{mn}(r) = \sum(-1)^{s} F(m,n,s,r) \quad (4)$$

And

$$F(m,n,s,r) = \frac{(m+s)!}{s!(\frac{m+|n|}{2}s)!(\frac{m-|n|}{2}s)!} r^{m-2s} \quad (5)$$

Zernike moment can be very relevant in digit recognition. They owe their performances to their rotation and translation invariance properties. However, such invariance leads to some limitations in dissociating "6" and "9" digits. In this study, we will consider terminating points detector for the seek of non invariant descriptors.

Terminating points based features: The endpoint number is an estimation of terminating points (Lori *et al*., 1981). Figure 7 shows for illustration a handwritten digit "1" which endpoints are surrounded. Once endpoint extracted, the following descriptors can be computed.

The number of terminating points N_T: The endpoint number is closely related to topological characteristics of digits. Often, its is specific. Ever though the results can not be unique, it is always bounded and can be predicted.

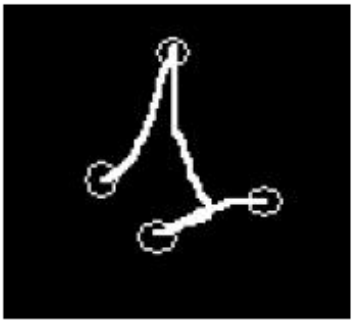

Fig. 7: Endpoints of digit "1"

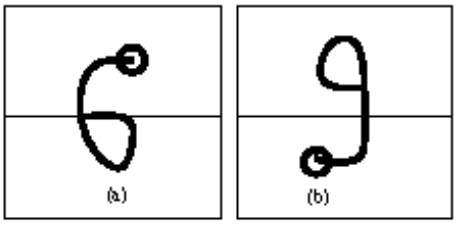

Fig. 8: "6" and "9" problem: (a) $N_L = 1$, (b) $N_L = 2$

The number of terminating point location N_L: Accurate location of the endpoints is important for reliable and robust digit recognition. Besides, location in the yapper or lower half of the image are invariant for some digits and can be used for dissociating "6" and "9" (Fig. 8).

Let X_i, Y_i, be coordinate of the i^{th} ends of the skeleton and X_c Y_c be coordinate of the half of the image. Since, digits 6 and 9 are the only which have a unique endpoint, we carry out this algorithm for distinguishing them from each other:

If (($N_T = 1$) and ($X_i > X_c$)
then $N_L = 1$
elseif (($N_T = 1$) and ($X_i > X_c$)
then $N_L = 2$
else $N_L = 0$
end

THE WHOLE NEURAL NETWORK

For each digit sample, the values 6th order of Zernike moments, M_z ($z = 1, ..., 7$), the terminating points number N_T and the location number N_L are computed. The description based on these features is used in the recognition step as neural network inputs. The applied network architecture is represented in Fig. 9.

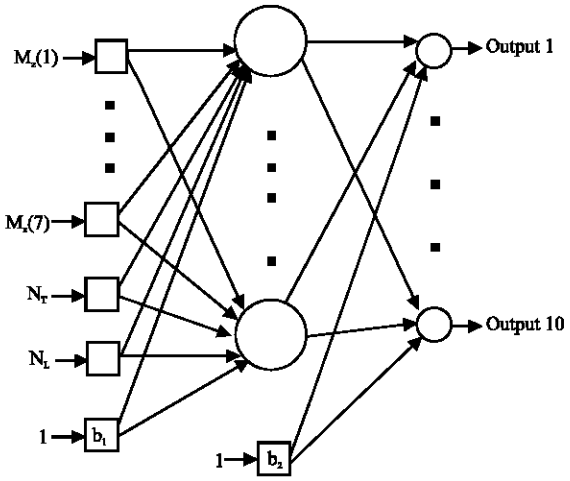

Fig. 9: The applied network architecture

This network was trained with the data base samples using the backpropagation algorithm.

The backpropagation training algorithm: The backpropagation training algorithm is basically a steepest descent method that searches for optimal weight values to minimize the error E_{rr} between the network output and the target. The procedure is the following:

- First guess an initial weight w^q, q = 0, where q denotes the current training iteration number.
- Find $\partial E_{rr}/\partial w$, which is the derivative of E with respect to w.
- Find the change of weights to give smaller E_{rr} by the steepest:

$$\Delta w^q = -\eta \frac{\partial E_{rr}(w)}{\partial w}\Big|_{w=w^q} \quad (6)$$

where is the learning rate.

- Update the weight w^q to w^{q+1} as:

$$w^{q+1} = w^q + \Delta w^q \quad (7)$$

- Repeat steps (ii) to (iv) until $\partial E_{rr}(w)/\partial w = 0$, which means that you are at the bottom of the valley, where is the solution.

For the output layer, the quantity $\partial E_{rr}(w)/\partial w$ in Eq. 6 can be easily calculated by taking the derivative of E_{rr} with respect to the weights w of output layer, while $\partial E_{rr}(w)/\partial w$ for the weights in the hidden layers requires the use of chain rule to backpropagate the error signal obtained in the output layer to the hidden layers.

Table 1: Generalization error for different Zernike orders with ten hidden layers

Number of Zernike moments	Zernike moments as inputs	The proposed feature vector
4	0.9103	0.532
6	0.7642	0.152
8	0.6844	0
10	0.3641	0

Table 2: Generalization error for different hidden layer number with Zernike order of six

Number of the hidden layers	Zernike moments as inputs	The proposed feature vector
5	0.8871	0.057
7	0.766	0.014
10	0.6301	0
20	0.2565	0

To be able to evaluate the effectiveness of the training, one can measure the relative error as following:

$$E = \frac{\sum(I_N\text{-Target})}{\text{Number of samples}} \quad (8)$$

where, I_N is the image resulting from network output and Target is the binary image. Indeed, using this pulse network can lead to very low error rates.

Generalization rate results: After training step, generalization error was evaluated for different feature and network conditions. Table 1 illustrates the generalization error for different Zernike orders with ten hidden layers. In case of Zernike moment feature based, for Zernike orders smaller than 6, the network error remains high, even though increasing the number of hidden layers to 20 (Table 2). Besides, Table 1 shows that considering endpoint based features give satisfactory results, from an order of Zernike moments of six. For this order, simulation results of the network error for different hidden layer numbers is illustrated in Table 2. From this table, we ca notice that increasing the network size beyond ten, can decrease significantly the error.

Indeed, reconstructed digits from 6 order Zernike moments illustrated in Fig. 10, show that it is hard distinguishing the difference. That's the use of endpoint features makes discrimination easier and reduces the network complexity.

Figure 11 and 12 show training and generalization error evolution for different Zernike orders and different Hidden Layers, respectively. Accordingly, an optimum choice of Zernike order can be done ensuring a compromise between low network error and compact network.

In the present study, the proposed feature vector will be implemented as a pulse mode neural network inputs for handwritten recognition.

Fig. 10: Reconstruction of digits from their sixth order Zernike coefficients

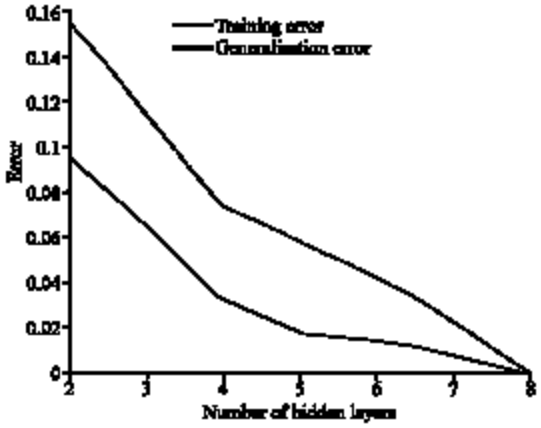

Fig. 11: Evolution of training and generalization errors by varying the hidden layers number with a Zernike order of six

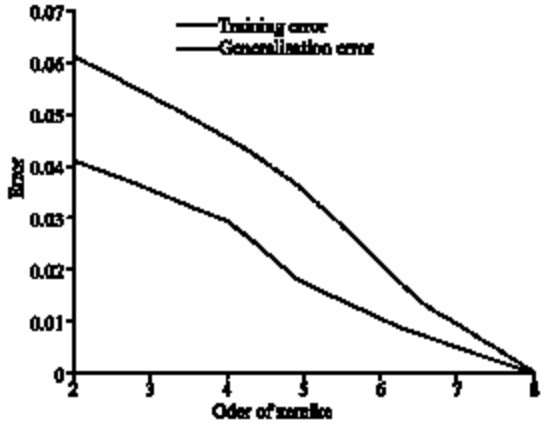

Fig. 12: Evolution of training generalization error by varying the order with ten hidden layers

HARDWARE IMPLEMENTATION ON AS VIRTEXII PLATFORM

In this research, we present the proposed neural network based handwritten recognition with floating point format operations.

Fig. 13: Floating point binary format

Pulse mode neural network bloc architectures

Representation of signals with floating point format: Floating point is a numeric interpretation system in which a string of digits (or bits) represents a real number. Floating-point format is used here to represent the synaptic weights in the see of unlimited range operations, it is illustrated in Fig. 13.

The adopted format is as follows:

$$(\text{Sign}) * 2^{\text{exponent-shift}} * 1, \text{mantissa} \quad (9)$$

With

$$\text{shift} = 2^{n-1} - 1 \quad (10)$$

where, n is the number of bits allocated to the exponent.

Synapse unit: In the synapse unit, neuron output is multiplied by a synaptic weight. As the proposed network uses frequency to represent the signal levels, the multiplier is replaced by a frequency converter using floating point operations.

Exponent (W_{exp}) has p bit and mantissa (W_{mant}) has q bit length. Block diagram of the synapse unit is depicted in Fig. 14.

Figure 15 shows the block diagram of the proposed synapse multiplier. Each weight is divited into n_w bits which are multiplied by the corresponding pulse input using simple AND gates. These outputs are concatenated to have a n_w-bit signal representing the weight multiplication w_i, I_i. The total sum of the weight values is the product of the signal level by the weights.

Neuron unit: The sigmoid function is by far the most common form of transfer function in Neural Networks. However, the design of this function on platform is expensive. In deed several possibilities exist to design the sigmoid function. We can use its polynomial approximation. This solution can be expensive in computing time and in silicon area. To avoid this complexity, we can truncate it to an arbitrary precision by a judicious piecewise linear approximation (Haykin, 1999) or use a lookup table placed in a ROM (Peng *et al*., 1999).

The nonlinear activation function is approximated by a sum of three weighted ramp functions. It is given by the following equation.

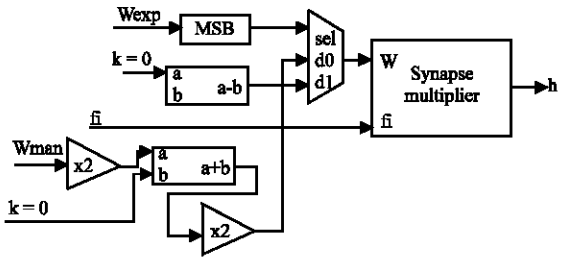

Fig. 14: Synapse unit

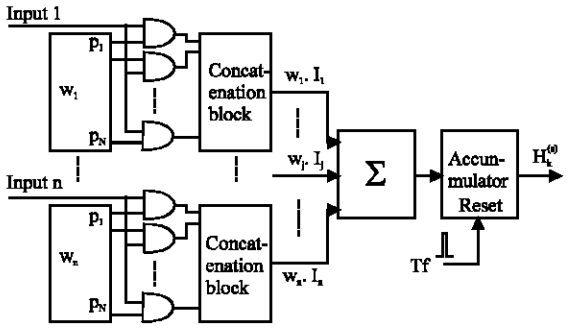

Fig. 15: Block diagram of the proposed synapse multiplier

$$f(H) = \frac{3}{6} f_{p0}(H) + \frac{2}{6} f_{p1}(H) + \frac{1}{6} f_{p2}(H) \qquad (11)$$

where, H is the internal potential in the neuron which is the product of the sum of weight values h and the average frequency F.

$$H = h.F \qquad (12)$$

Each ramp function can be expressed by the following equation.

$$f_{pj}(H) = \begin{cases} 1 & \text{if} \quad H > p_j \\ \dfrac{H}{(2p_j)} + 5 & \text{if} \quad -p_j \le H \le p_j \\ 0 & \text{otherwise} \end{cases} \qquad (13)$$

where, p_j (j = 0, ..., 2) is the control parameter of the slope of the ramp function.

The block diagram of a single ramp function neuron is depicted in Fig. 16. Figure 17 shows the transition of the register value and neuron output.

Each period T_f, the sum of input weight values from synapse multipliers $H_k^{(s)}$ is fed to the register where it is accumulated. When the content of the register R is positive, p_j is subtracted from it, otherwise it is added. The

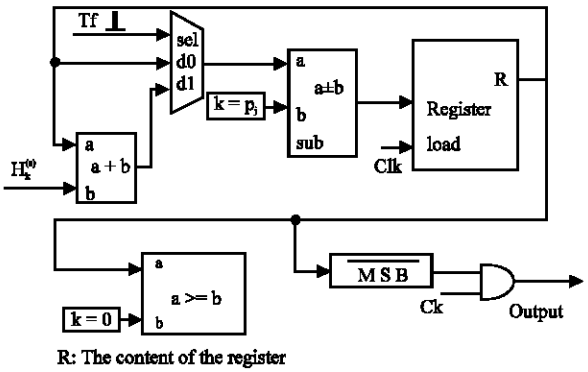

Fig. 16: Ramp function neuron

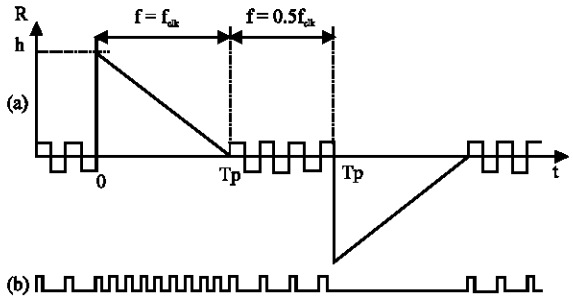

Fig. 17: (a) Evolution of the register in the ramp function. (b) Neuron output

output pulse of ramp function is generated with a frequency equal to the frequency of the clock f_{clk}. Afterwards, the content of the register alternate positive and negative and the gradient of the change is p_j.

The time spent by the register to approach zero is T_p, later on, the frequency of the output becomes 0.5 * f_{clk}. Hence the average frequency of the neuron ramp function is given by the following equations:

$$f_{pj}(H) = \begin{cases} \dfrac{Tp + (T_f - T_p)*0.5}{T_f} & \text{if} \quad H > 0 \\ \dfrac{(T_f - T_p)*0.5}{T_f} & \text{if} \quad H < 0 \end{cases} \qquad (14)$$

j = 0,..., 2.

where T_f and T_p are given by:

$$T_f = \frac{1}{F} \qquad (15)$$

$$T_p \frac{|h|}{p_j} \qquad (16)$$

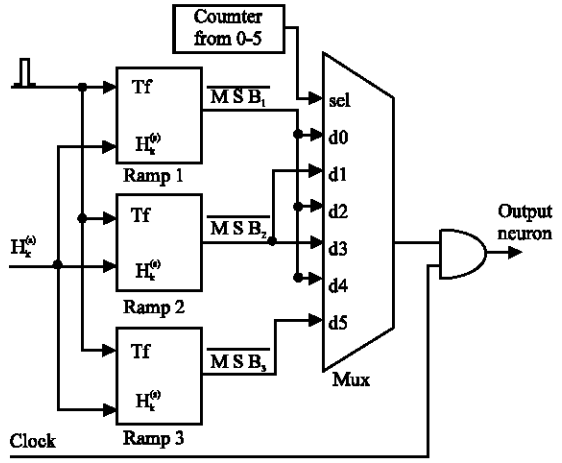

Fig. 18: The neuron with piecewise-linear activation function

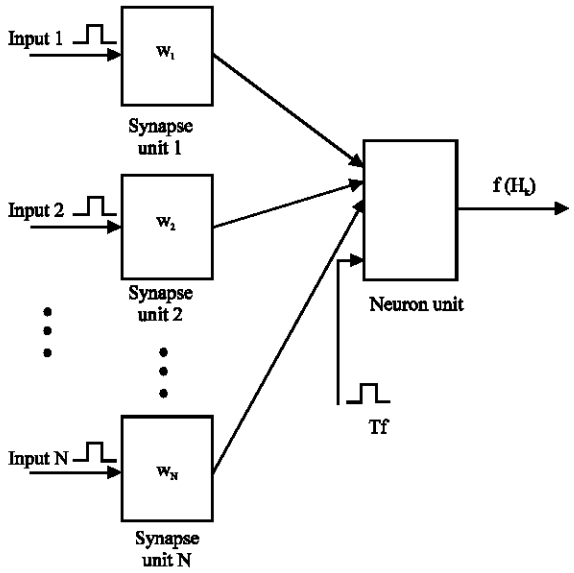

Fig. 19: Block diagram of one neuron unit connected with N synapse units to characterize the activation function

By substituting Eq. (15), (16) and (12) in (14), we obtain an activation function with a shape of ramp function described in Eq. (13).

To obtain the weighted sum of three ramp functions, a six to one multiplexer is used. Figure 18 shows the block diagram of the neuron to get the proportions indicated in Eq. (11).

For 6 clock cycles of the counter, the first ramp function is selected three times, the second ramp is selected twice and the third one is selected once.

To characterize the neuron's activation function, we have implemented one neuron unit connected with a certain number than ten of synapse units as shown in Fig. 19. The weight values are expressed in 12-bit signed fixed-point format. We use frequency converters to generate pulsed input signals from numeric values expressed in six-bit signed fixed-point format (nb_{in} = 6). Both weight values and input frequencies are randomly set. The period T_f that should be left for correct propagation of signals is expressed by (17).

$$T_f = 2^{nb}in \quad (17)$$

RESULTS

Activation function characterization: Unlike stochastic neuron, the ramp function neuron isn't affected by the number of input signals and weight range. Figure 20 represents the activation function of the neuron with 6 inputs. This figure shows measurement results of ramp function neuron with different parameters p_j (j = 0, ..., 2).

The same configuration shown in Fig. 19, excepting the ramp function neuron which is replaced by the neuron unit, is used to plot the piecewise linear activation function for more control of the neuron.

Figure 21 shows the slopes of the piecewise activation function with different number of inputs. The activation function curves are very close to theoretical expressions of sigmoid function.

Synthesis results: The proposed multilayer neural network was implemented on FPGA VirtexII platform. The precision of the synapse weight values is expressed in 9 bit precision.

Table 3 describes the device utilization summary after design synthesis. We can note that the proposed architecture uses less than 50% of hardware resources of the target FPGA so it can be applied in larger networks. Compared to conventional architectures, a much more compactness is obtained (Krid and Masmoudi, 2005).

Table 4 summarizes the device timing. The maximum frequency that can be reached is 101.256 MHz and the minimum period is 9.876 ns. Since, the proposed architecture is undertaking in fully parallel operations, this value decreases in proportion to the network scale.

For validation and test, a series of digits data base was used. Each digit for the data base has a generalization error that doesn't exceed 4.34%.

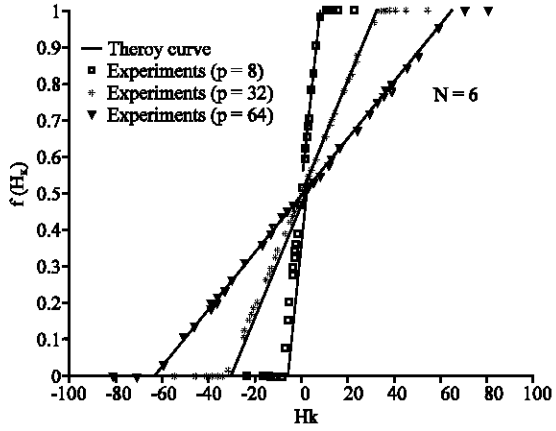

Fig. 20: Characterization of the neuron with one ramp functionvconnected to six synapse units for different values of parameter p

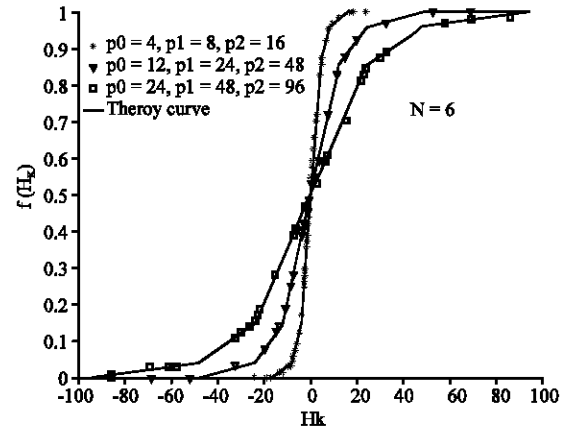

Fig. 21: Characterization of the neuron unit with six input signals for different combinations of p_i

Table 3: Device utilization summary

Number of external IOBs	8 out of 248	3%
Number of RAMB16s	8 out of 44	18%
Number of SLICEs	839 out of 4928	17%
Number of BUFGMUXs	1 out of 16	6%

Table 4: Timing summary

Minimum period	9.876 ns
Maximum Frequency	101.256 MHz
Minimum input arrival time before clock	2.911 ns
Maximum output required time after clock	9.094 ns

CONCLUSION

New digital architecture of pulse mode neural network applied to handwritten digit recognition with floating point operations was proposed. This approach is based on Zernike moments and endpoint descriptors. The improved architecture proposes an efficient method to accurately compute feature vectors for best recognition. The hardware complexity is reduced by detecting the number of endpoints and the number of terminating points location. The activation function uses floating point operations while providing the same performance as conventional architectures, leading to unlimited weight range operations. Feasibility of the proposed architecture was verified by computer simulations. Moreover, the proposed architecture was implemented on a Virtex II FPGA platform. Simulation results show that the proposed neural network has good learning performances.

ACKNOWLEDGEMENT

The authors would like to acknowledge the support from the Sfax Engineering School (ENIS) and thank the Computers Imaging Electronics and Systems (CIELS) group for the various help.

REFERENCES

Damak, A., M. Krid, D.S. Masmoud and N. Derbel, 2006. FPGA Implementation of Programmable Pulse Mode Neural Network with on Chip Learning Iinternationnal Conferance DTIS.

Haykin, S., 1999. Neural Networks, A comprehensive Foundation. Prentice Hall. 2nd Edn.

Hikawa, H., 1999. An efficient pulse mode multilayer neural network. Proc. IEEE. ISCAS, pp: 367-370.

Hikawa, H., 1999. Frequncy-Based Multilayer Neural Network with on chip learning an Enhanced Neuron characteristcs. IEEE. Trans. Neural Networks, 10 (3): 545-553.

Hikawa, H., 2000. Pulse Mode Multilayer Neural Network with Floating Point presentation. ISCAS 2000-IEEE International Symposium on Circuits and Systems, Geneva, Switzerland.

Ikenaga, T. and T. Ogura, 1998. A DTCNN universal machine based on highly parallel 2-D cellular automata CAM. IEEE. Trans. Circuits Syst. I, 45: 538-546.

Khotanzad, A. and Y.H. Hong, 1990. Invariant image recognition by zernike moments. IEEE. Trans. Pattern Anal. Machine Intell., 12: 489-498.

Krid, M. and D.S. Masmoudi, 2005. FPGA Implementation of a Feedforward Neural Network 3rd International Conference on Systems, Signals and Devices Sousse, unisia.

Krid, M., A. Damak and D. Sellami Masmoudi, 2006. FPGA Implementation of Programmable Pulse Mode Neural Network with on Chip Learning for signature application Iinternationnal Conferance ICECS.

Krid, M., D.S. Masmoudi and M. Chtourou, 2005. Hardware implementation of bfnn and rbfnn in FPGA technology: Quantization issues Internationale Conference, ICECS.

Lehmann, C., M. Viredaz and F. Blayo, 1993. A generic systolic array building block for neural networks with on chip learning. IEEE Transaction Neural Networks, Vol. 4.

Lori F. Lamel, 1981. Student Member, IEEE, Lawrence R. Rabiner, Fellow, IEEE, Aaron E. Rosenberg, Member, IEEE and Jay G. Wilpon, An Improved Endpoint Detector for Isolated Word Recognition IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-29, No. 4.

Maeda, Y. and T. Tada, 2003. FPGA implementation of pulse density neural network with learning ability using simultaneous perturbation. IEEE. Trans. Neural Networks, 14 (3): 688-695.

Marchesi, M. and G. Orlandi, 1993. Fast Neural Networks Without Multipliers. IEEE. Acoust. Speech, 4 (1): 53-62.

Martincigh, M. and A. Abramo, 2005. A new architecture for digital stochastic pulse mode neurons based on the voting circuit. IEEE. Trans. Neural Networks, 16 (6): 1685-1693.

Moon, G., M.E. Zaghloul and R.W. Newcomb, 1992. VLSI Implementation of Synaptic Weighting and Summing in pulse Coded Neural-Type Cells. IEEE. Tmns. Neuml Networks, 3 (3): 394-403.

Oulhadj, H., J. Lemoine, E. Petit and H. Wehbi, 1999. Combinaison d'algorithmes pour la reconnaissance des chiffres et des lettres batons dans un environnement multiscripteur decriture courante mixte. Vision Interface, Trois-Rivires, Canada.

Peng, W., M. Ter Brugge, J. Nijhuis and L. Spaanenburg, 1999. Mapping a Trained Neural Network on Hardware. International Computer Science Conference ICSC IIA/SOCO.

Reyneri, L.M., 1995. A performance analysis of pulse stream neural and fuzzy computing system. IEEE. Trans. CAS, 42 (10): 624-660.

Shutler, J.D. and M.S. Nixon, 2006. Zernike velocity moment for sequence-based description of features. Science Direct.