Journal of Computer Science 2 (4): 355-362, 2006
ISSN 1549-3636

Digital Hardware Implementation of a Neural System Used for Nonlinear Adaptive Prediction

[1]Hassène Faiedh, [1]Chokri Souani, [2]Kholdoun Torki and [1]Kamel Besbes
[1]Laboratoire de Microélectronique et Instrumentation µEI, Faculté des Sciences de Monastir, Boulevard de l'environnement, 5019 Monastir, Tunisie
[2]Techniques de l'Informatique et de la Microélectronique pour l'architecture d'ordinateurs (TIMA) Institut National Polytechnique de Grenoble (INPG), 46 Avenue Félix Viallet, 38031 Grenoble, France

Abstract: Neural networks have been widely used for many applications in digital communications. They are able to give solutions to complex problems due to their nonlinear processing and their learning and generalization. Neural networks are one of the key technologies for the communication domain and accordingly a special effort may be expected to be paid to real time hardware implementation issues. In this study, it is proposed a digital hardware implementation of a neural system based on a multilayer perceptron (MLP). The neural system is used for the nonlinear adaptive prediction of nonstationary signals such as speech signals. The implemented architecture of the MLP is generated using a generic elementary neuron (EN). The polynomial approximation method is used to implement the sigmoidal activation function. The back-propagation algorithm is used to implant the prediction task. The circuit implementation architecture is detailed, for achieving real-time prediction for speech signals. The designed ASIC circuit includes a neural network block, an on-chip learning block and a memory used for storing the synaptic weights for updating.

Key words: Digital hardware implementation, artificial neural networks, multilayer perceptron, on-chip learning, nonlinear adaptive prediction

INTRODUCTION

Many physical signals, such as speech, are generated from a nonlinear mechanism and have statistically nonstationary properties, which make the task of their prediction difficult. Artificial neural networks have been widely used as powerful tools for modelling nonlinear dynamical systems. They are also able to give solutions to complex problems in digital communications due to their nonlinear processing, parallel distributed architecture, capacity of learning and generalisation and the efficient hardware implementation. A neural network is well suited for the nonlinear prediction of nonstationary signals by virtue of the distributed nonlinearity built into its design and the ability of the network to learn from its environment.

Digital implementation of neural networks on configurable systems was presented by[1,2]. Also, several commercial hardware solutions that can be used to implement neural circuits have reached the market[3]. The learning algorithm implementation remains the main difficulty when an autonomous system is planned regarding the running frequency. The implementation of the nonlinear activation function of neurons and its derivative used by the learning algorithm, is often solved by a linear approximation[4-6] but no implementation method has emerged as a universal solution[7].

In this study we propose a hardware implementation of a neural system, used for prediction in time-series. An adequate and optimal architecture is used for the implementation of the learning algorithm. As a final result, we propose a digital hardware implementation of a neural prediction circuit on digital ASIC.

Problem position – brief presentation: Neural networks are able to give solutions to complex problems in digital communications due to their nonlinear processing, parallel distributed architecture, capacity of learning and generalization and efficient hardware implementations[8]. In this paragraph we describe a neural system for the nonlinear adaptive prediction of non-stationary signals and demonstrate its application to a speech signal[9,10]. The neural network used here is a multilayer perceptron (MLP) trained with the backpropagation algorithm. This structure is not recursive; it permits days reduction of training and gives a big gain in the time of the hardware implementation.

The prediction system: The system is modeled through the use of a feedforward multilayered neural network fitted with tapped delay lines at its input. Figure 1 shows a schematic diagram of the neural

Corresponding Author: Chokri Souani, Laboratoire de Microélectronique et Instrumentation µEI, Faculté des Sciences de Monastir, Boulevard de l'environnement, 5019 Monastir, Tunisie, Tel: +216 73 500 280 Fax: +216 73 500 278

network and its environment. The purpose of the design is to filter a set of samples Y, of the input signal S that is represented by equation:

$Y = [S(t), S(t-1), S(t-2), S(t-3), \ldots]^T$.

The subscript T denotes transposition. These samples give the past of the signal at the variation of a discrete time t. The aim of the operation is to produce a prediction $S^P(t+1)$ of the signal one step into the future. $S^P(t+1)$ value is used to update the weight values in the neurons network. The prediction error is injected into the learning bloc.

An example of a similar predictor was given by[9]. The pipelined recurrent neural network (PRNN) gives satisfactory results but is relatively complex for hardware implementation. A non-recurrent neural network may be less complex and easier to implant the entire prediction system on silicon chip.

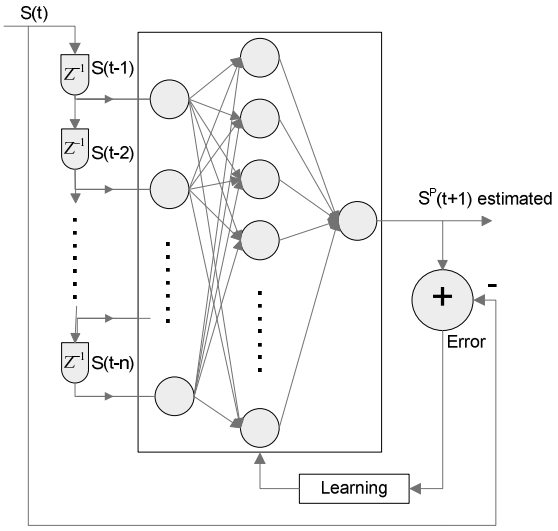

Fig. 1: The neural network and its environment

The non-recurrent neural network: The structure of the neural network used in the design consists of a non-recurrent fully connected multilayer perceptron (MLP). The proposed MLP architecture presents three layers. The first layer is the input layer composed by *p* neurons. The hidden layer is the second and is composed by *q* neurons used for intermediate calculus. The output layer, the third one, is composed by a unique neuron and calculates the predicted value.

Let W1 and W2 denote the synaptic weight matrix respectively for the first and second layers.

$$W1 = \begin{bmatrix} W1(1,1) & \cdots & W1(1,p) \\ \vdots & \cdots & \vdots \\ W1(q,1) & \cdots & W1(q,p) \end{bmatrix} \text{ and } W2 = \begin{bmatrix} W2(1,1) \\ \vdots \\ W2(1,q) \end{bmatrix}$$

The activation function for every neuron in each layer is the sigmoid binary function described by $f(x) = 1/[1+\exp(-x)]$, where x is the internal potential of each neuron.

Let X1 the output vector of the second layer, calculated by: $X1 = f(W1 \cdot Y)$. Assume X2 the output of the third layer expressed as: $X2 = f(W2 \cdot X1)$.

A learning algorithm calculates, at each time step, the weight correction factors ΔW1 and ΔW2 in order to update the weight matrixes W1 and W2. The error function is calculated by comparison between the estimated and the real value of the sample, $e(t) = S(t) - S^P(t+1) = S(t) - X2$.

The prediction task of a nonstationary time series, such as speech signals, needs a continuous learning. Owing to the fact that we cannot estimate the neurons error in the hidden layer, we chose the backpropagation learning algorithm detailed in[11] to correct the synaptic weights. The new matrixes are calculated according to the following equations: $W1_{new} = W1 + \Delta W1$ and $W2_{new} = W2 + \Delta W2$.

In our case the backpropagation algorithm used for training the neural network is performed by the equations:

$\Delta W_{k,h}^{(3)} = -a \cdot \delta_k^{(3)} \cdot y_h^{(2)}$: update weights in the third layer

$\Delta W_{k,h}^{(2)} = -a \cdot \delta_k^{(2)} \cdot y_h^{(1)}$: update weights in the second layer

$\delta_K^{(2)} = [\ \delta_i^{(3)} \cdot W_{i,k}^{(3)}\ ] \cdot f'(v_k^{(2)})$: output error of the second layer

$$v_k^{(2)} = \sum_{m \in layer1}^{P} W_{k,m}^{(2)} \cdot y_m^{(1)}$$: the potential of neurons

$W_{k,h}$ is the synaptic weight between the neuron k of previous layer and the neuron h of the considered layer. "δ" represents the output error of a neuron. "a" is the learning rate and *v* is the potential of each neuron before activation. The numbers between parentheses represent the layer number and "y" represents the neuron's output.

SIMULATION RESULTS

Three different speech signals, denoted by S1, S2 and S3 were used to test the nonlinear predictor. These signals are registrations sampled at 8 KHz and coded on 8 bits. The amplitude of the signals is normalized to be in the definition domain of the function *f*.

The numbers *p* and *q* of neurons in the first and second layer are determined by an optimization procedure in order to minimize the square error function *E* expressed by:

$$E = \frac{\sum_{n=1}^{Number\ of\ samples} e^2(n)}{Number\ of\ samples}$$

Optimal parameters obtained are p=2 and q=12[12]. Figure 2-4 represent the temporal representation of the various signals. The curve with stars '☆' represents the real signal stacked with the predicted signal represented with cross 'x'. The error curve represents the difference between signals with a continuous line.

The error signal is of reduced amplitude. There is a very light gap between the signals. This gap is due to the procedure of weights update which makes late by re-injection of the error signal.

That is why the maximal value of the error is not significant. Moreover this value does not exceed 31%. The mean value of the error is lower than 4% and the square error is lower than 0.32 %. Table 1 shows the mean, the mean square and the maximum value of the prediction error.

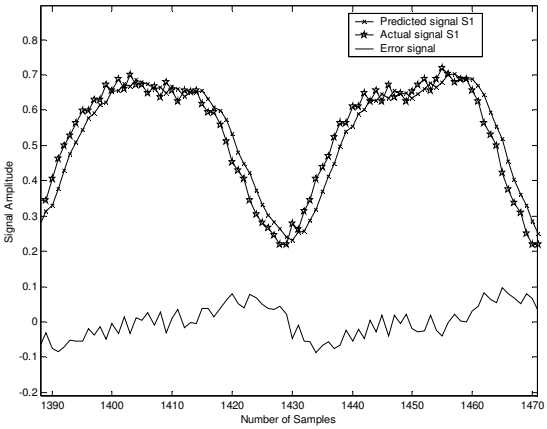

Fig. 2: Nonlinear prediction of the speech signal S1

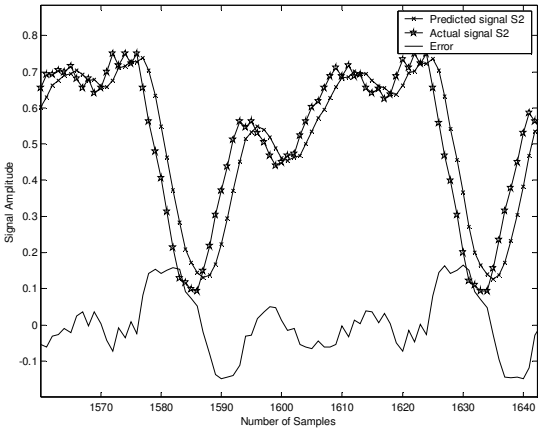

Fig. 3: Nonlinear prediction of the speech signal S2

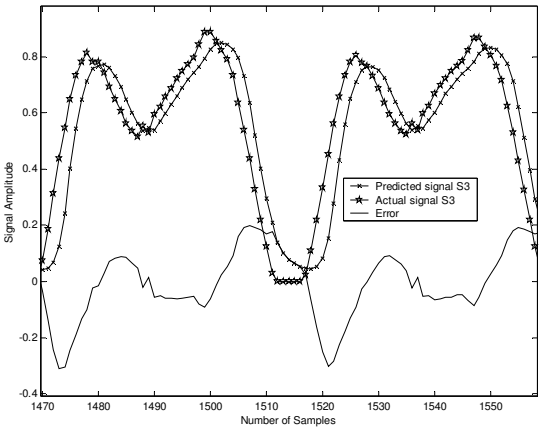

Fig. 4: Nonlinear prediction of the speech signal S3

Table 1: The prediction error

Signal	Mean absolute Error $\|e\|$	Mean (squared error) E	Max. Absolute Error $\|e\|$
S1	2.2%	$0.8\ 10^{-3}$	15%
S2	3.4%	$2.4\ 10^{-3}$	22%
S3	3.1%	$3.2\ 10^{-3}$	31%

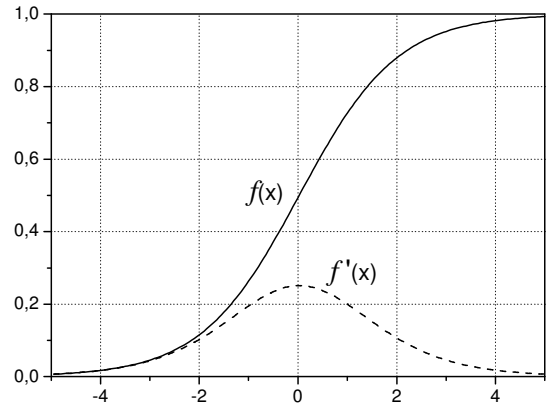

Fig. 5: f is the sigmoid function and f ' is its first derivative

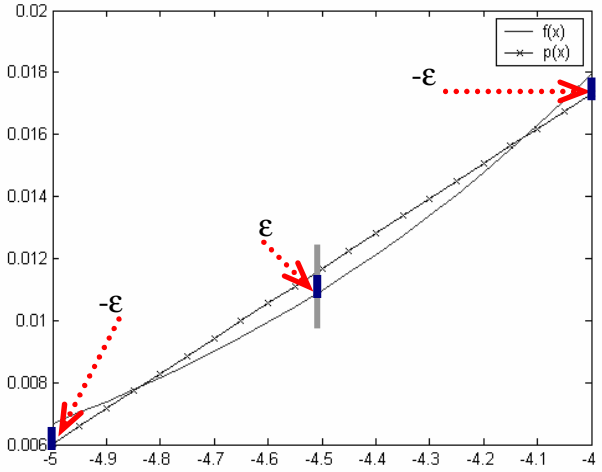

Fig. 6: Polynomial approximation of the function f on [-5, -4]

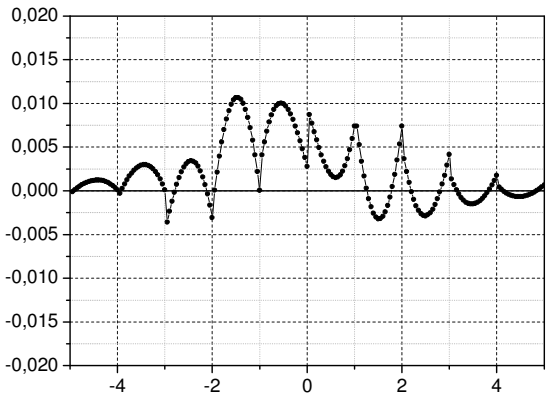

Fig. 7: Representation of the approximation error

Implementation of the activation function: For hardware implementation, many constraints must be

considered, such as the nonlinearity of the activation function, the silicon area and the time delay. The most popular activation function is the sigmoid, often used with gradient-descendent type learning algorithms[11,13-16] as represented in Fig. 5. There are different possibilities to implement this function such as look-up tables or piecewise linear approximation. Moreover, by using the gradient descent algorithm, it is necessary to use the first derivative of the activation function described by:

$$f'(x) = f(x)\left[1 - f(x)\right]$$

The operation realized by an artificial neuron unit in a multilayer perceptron is described by:

$$\text{Output} = f\left(\sum_{i=1}^{n} w_i \cdot x_i\right)$$

where x_i is the input of the neuron, w_i is the synaptic weight related to the considered input, f is the neuron activation function and n is the total number of all inputs.

The implementation of the neuron's nonlinear activation function and their derivatives used by the learning algorithm, is often solved by a piecewise linear approximation [4,5,7,17-20]. However, no implementation method has emerged as a universal solution. For hardware implementation, the efficiency criteria for a successful approximation are the achieved accuracy, speed and area resources. If the circuit is on-chip learning, the multiplier used by the learning algorithm (or used by the neuron to multiply the inputs with weights) could also be used in a time-sharing manner for computing an approximation of the sigmoid function. In this paragraph, we propose a polynomial approximation of the sigmoid activation function and its derivative used in artificial neural networks.

Preliminary study: Let P_N the set of polynoms of degree less or equal to integer N. Let p* an element of P_N, the approximate of the function f in the interval [a, b]. It is important to note that the Taylor development of the function f for a given point in the interval is in general not satisfactory because the Taylor development gives a local and not a global approximation.

Table 2: Polynomial approximation of the sigmoid function f

Interval	Polynom $P_1(x)$
[-5, -4]	0.01129·x + 0.06248
]-4, -3]	0.02943·x + 0.13404
]-3, -2]	0.07177·x + 0.25602
]-2, -1]	0.14973·x + 0.41285
]-1, 0]	0.23105·x + 0.49653
]0, 1]	0.23105·x + 0.50346
]1, 2]	0.14973·x + 0.58714
]2, 3]	0.07177·x + 0.74097
]3, 4]	0.02943·x + 0.86595
]4, 5]	0.01129·x + 0.93751

The following two theorems are fundamental results[21-23]. The first one is obtained thanks to Weierstrass and the second to Chebyshev. Let's find an approximation of a continuous function f on interval [a, b].

Using the approximation theorem announced by Weierstrass, it is showing that: for any ε>0, there exist a polynom P such that: $\text{Sup}_{[a,b]} |f(x) - P(x)| \leq \varepsilon$.

Meaning that, any continuous function can be approximated uniformly by a polynom. But, it gives us no information about the polynom's degree. Bernstein[23] announces that the polynom degree can be as large as we want.

Now we may use the Chebyshev theorem: P^*_N is the best uniform approximation polynom for f on [a, b], in the set of polynoms of degree less than or equal to N, if and only if are existing N+2 points x_i such that:

$a \leq x_0 < x_1 < x_2 < \ldots < x_{N+1} \leq b$

verifying:

$P^*_N(x_i) - f(x_i) = (-1)^i [P^*_N(x_0) - f(x_0)] = \pm \text{Sup}_{[a,b]} |P^*_N(x) - f(x)|$

Sigmoid function approximation: The sigmoid is a continuous function and strictly monotonous on]-∞, +∞[. In practice, we may consider that this function tends to 0 on]-∞, -5] and to 1 on [+5, +∞[. For complexity reasons, we consider only polynoms of degree equal to one. We apply the preceding theorems to find the best approximation of the sigmoid function on the intervals [-5, -4] to[4, 5] by a step of one. It is supposed that the function is constant on]-∞, -5] and on [+5, +∞[.

Let's present an example of approximation procedure within the interval [-5, -4]. The same procedure was applied to find the approximate of the function on the rest of the interval fragments.

Let $P_1(x) = a \cdot x + b$ be the approximation of f on [-5, -4]. The maximal error ε is reached in three points on the interval. Furthermore, the convexity of the function f in [-5, -4] implies that the 2 extreme points with the greatest error are the edges of the interval (Fig. 6).

Let's define α as the third point where this maximal error is reached, we have:

$f(-5) - p_1(-5) = -\varepsilon$,

$f(\alpha) - p_1(\alpha) = \varepsilon$,

$f(-4) - p_1(-4) = -\varepsilon$.

As the error is maximal at α point, then: $f'(\alpha) - p_1'(\alpha) = 0$.

We deduce easily the values: a = 0.01129, α = - 4.5, b = 0.06248 and ε = 0.0006.

This means that when the degree of the polynom is one, the function f is approximated on [-5, -4] by the polynom $P_1(x) = 0.01129 \cdot x + 0.06248$ and the maximum error (ε = 0.0006) is reached at the points -5, -4.5 and -4.

We apply the same procedure for the rest of the intervals. The detailed piecewise approximation results of the sigmoid function f are shown in Table 2.

Table 3: Average and maximum errors of the sigmoid function approximations

Approximation	Domain	Mean ε	Max ε	Maximum
A low-based approximation [17]		2.47%	4.90%	
Approximation of Alippi and Storti-Gajani [18]		0.87%	1.89%	
PLAN Approximation [19]		0.59%	1.89%	
CRI Approximation [5], q = 0		2.41%	11.9%	11.9%
CRI Approximation [5], q = 1	[-8,8[	1.20%	3.78%	
CRI Approximation [5], q = 2		0.92%	2.45%	
CRI Approximation [5], q = 3		0.85%	2.06%	
Approximation of Tommiska [7], sig_336p		0.33%	0.77%	
Approximation of Tommiska [7], sig_337p		0.17%	0.39%	
Our approximation	**[-5,5[**	**0.20%**	**1.11%**	**1.11%**
Approximation of Zhang et al. [4]		0.77%	2.16%	2.16%
Approximation of Tommiska [7], sig_235p	[-4,4[	0.69%	1.51%	
Approximation of Tommiska [7], sig_236p		0.40%	0.77%	

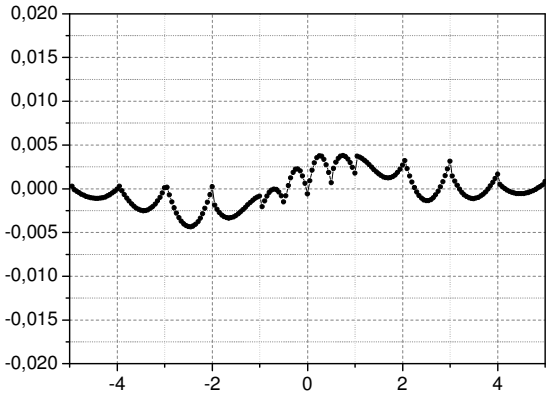

Fig. 8: Representation of the derivative approximation error

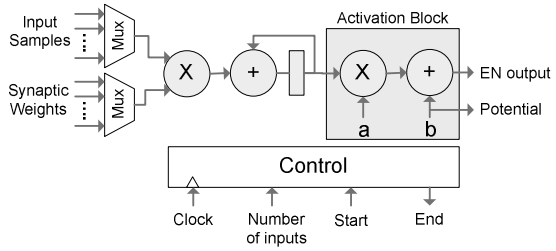

Fig. 9: General architecture of the artificial digital EN

The error approximation is represented on Fig. 7. The mean error is 0.2%. The maximum approximation error is 1.11%. Because of the error limitation, we accept this approximation.

A comparison between the sigmoid approximation result described in this work and others same approximations is resumed in Table 3.

By comparison, we can see that although the used intervals of cut are different, the maximal error for our approximation always remains lower than that found by the other authors.

Approximation of the first derivative function: By application of the same technique, the function $g(x)=f'(x) = f(x)\,[1- f(x)]$ can be approximated by a polynom of degree one. The approximate function $P_1(x) = a\cdot x + b$ is found at each fragment of the interval.

On the interval [-5, -4] for example, we obtain: a = 0.01101, α =-4.5, b = 0.06107 and ε = 0.0006.

Table 4: Polynomial approximation of the derivative function f '

Interval	Polynom $P_1(x)$
[-5, -4]	0.01101·x + 0.06107
]-4, -3]	0.02751·x + 0.12623
]-3, -2]	0.05981·x + 0.22213
]-2, -1]	0.09161·x + 0.28729
]-1, -1/2]	0.07678·x + 0.27442
]-1/2, 0]	0.02999·x + 0.25175
]0, 1/2]	-0.02999·x + 0.25175
]1/2, 1]	-0.07678·x + 0.27442
]1, 2]	-0.09161·x + 0.28729
]2, 3]	-0.05981·x + 0.22213
]3, 4]	-0.02751·x + 0.12623
]4, 5]	-0.01101·x + 0.06108

The detailed approximation results of the first derivative function *f '* are shown in Table 4. The error of derivative approximation is represented on Fig. 8. The maximum error is inferior to 0.5%. This error is very limited and the approximation is retained. We can say that this function is faithfully reproduced.

Digital implementation: The principal advantage of a digital implementation over the analogical one is the flexibility of the design flow. In this paragraph, we describe the digital implementation of the predictor system. The main parts of the target ASIC are: the neural network and the learning algorithm. A memory is used to store the synaptic weights for updating. A controller synchronizes all the parts of the circuit. The neural network system is based on a generic elementary neuron (EN).

The elementary neuron: Using HDL description of one EN, we generate an architecture composed of a MAC unit, multiplexers and the sigmoid calculus block which is composed of a multiplier and an adder. The general architecture of the digital artificial neuron is shown in Fig. 9.

A local control unit allows the synchronizing between different blocks. A "Start" signal triggers the calculation process by activating the EN processes.

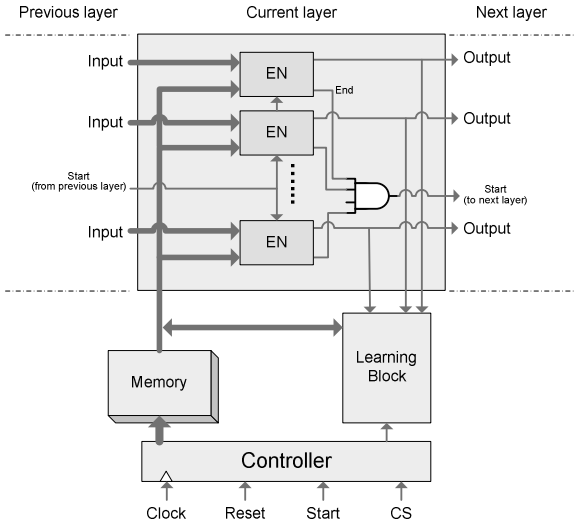

Fig. 10: Using the generic elementary neuron, we may construct any layer

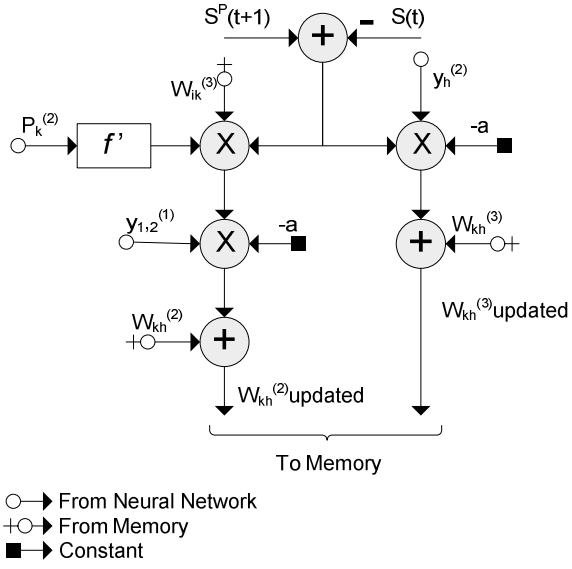

Fig. 11: The learning block

'End' signal indicates the end of the process. Except the first layer, each neuron on each layer has two inputs bus: the input samples and the synaptic weights. The EN outputs will be connected to the next neurons of the next layer. It communicates the calculated value and its internal potential. The neuron's potential value will be used by the learning block.

Neural network Implementation: The EN's architecture is then used to build the entire multilayer perceptron (MLP). Each layer is connected to the next one. Each layer contains the appropriate number of elementary neurons (p and q). A global control unit commands and controls all the neurons and the circuit blocks in the design. The signal "Start" triggers the process. When all the neurons of one layer finish, by raising to high all the "End" signals, the next layer is triggered. A logic "And" gate controls this process. Synaptic weights are stored in an on-chip RAM memory. The output neurons and the synaptic weights are communicated to the learning block.

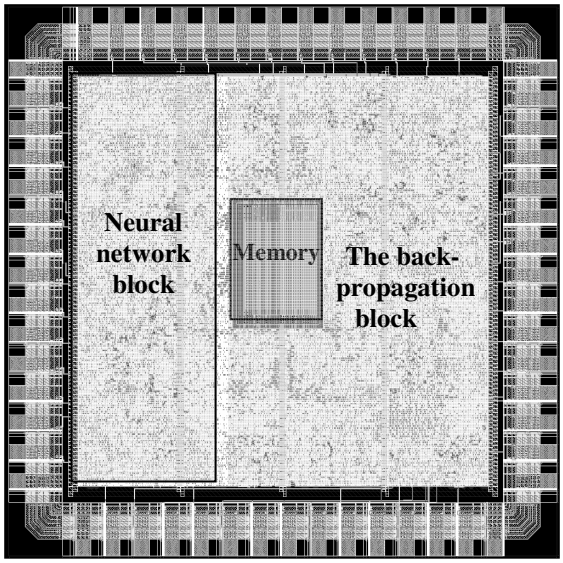

Fig. 12: Layout of the speech prediction ASIC

The memory used is a static Single Port RAM. It is used a 0.35µm pre-designed circuit. The delay time is 4 ns. Figure 10 represents a neuron's network architecture with the generic EN.

Learning algorithm implementation: The prediction system needs a real time gradient-descendent algorithm type for learning. We chose the backpropagation algorithm[11] as expressed.

The learning algorithm used here is the well known backpropagation algorithm[11,13]. The arithmetic operations used to perform the equations are addition, subtraction and multiplication. This module, performing the learning algorithm, has the inconvenient of a relatively very slow circuit because of many arithmetic multiplications built on it. It is the critical part in the circuit. Figure 11 shows the detailed architecture of the learning block.

The learning block, reads the synaptic weights from the memory to perform the calculus. Then it stores the new synaptic weights in the memory to be reused next process for prediction. The block calculating f ' is a simple implementation of a linear function using a ROM. The ROM contains the constants needed for calculating the approximation per segment as described in Table 4.

Tests and circuit design: Neural networks systems are known to be able to extract and restore information from a wrong or disturbed data. This ability is due to the capability of the neural network to learn from the

environment and its distributed non-linearity built into its design.

The ASIC circuit was designed using 0.35µm CMOS technology. The circuit area is about 8 mm^2 and contains 250 thousand logic gates. Figure 12 shows the layout of the ASIC.

The minimum delay time is about 25 ns, specifying a maximum frequency of 40 MHz. The critical path is maximized on the learning block, where many arithmetic operations are processed. Circuit emulation shows that it is possible to reach 50 MHz. The procedure of prediction creates some mistakes in calculation, but the stability of the process is assured.

CONCLUSION

In this work, a neural system used for time series predictions has been designed and hardware implemented with a fully digital manner. The system implemented is an interconnection between a neural network, a memory, a controller and a learning module. The neural network part is built on an elementary generic neuron (EN). The device is on-chip learning. Considering the robustness of the ASIC implementation, we observe process stability with some little generated calculus errors, when we use frequencies greater than 40 MHz. The ASIC area is about 8 mm^2 using 0.35 µm technology.

REFERENCES

1. Hammerstrom, D.A., 1995. Digital VLSI Architecture for Real-World Applications. An Introduction to Neural and Electronic Networks, second edition, Academic Press, pp: 335- 358.
2. Ramacher, U., J. Beichter, W. Raab, J. Anlauf, N. Brüls, U. Hachmann and M. Wessling, 1991. Design of a 1st generation Neurocomputer: VLSI Design of Neural Networks. Kluwer Academic Publishers, Boston, MA, pp: 271-310.
3. Morgado, F.D., A. Antunes and A.M. Mota, 2004. Artificial neural networks: A review of commercial hardware. Engg. Appl. Artif. Intell., 17: 945- 952.
4. Zhang, M., S. Vassiliadis and J.G. Delgado-Frias, 1996. Sigmoid generators for neural computing using piecewise approximations. IEEE Trans. Computers, 45: 1045-1049.
5. Basterrextea, K., J.M. Tarela and I. Del Campo, 2001. Approximation of Sigmoid Function and the Derivative for Artificial Neurons, Advances in neural networks and applications. WSES Press, Athens, pp: 397-401.
6. Faiedh, H., Z. Gafsi, K. Torki and K. Besbes, 2001. Digital hardware implementation of sigmoid function and its derivative for artificial neural networks. IEEE Proc. 13th Intl. Conf. on Microelectronics, ICM'2001, Rabat, Morocco, Oct. 29-31, pp: 189-192.
7. Tommiska, M.T. Efficient digital implementation of the sigmoid function for reprogrammable logic. IEEE Proc. Computer and Digital Techniques, 150: 403- 411.
8. Ibnkahla, M., 2000. Applications of neural networks to digital communications-A survey. Signal Process., 80: 1185- 1215.
9. Haykin, S. and L. Li, 1995. Nonlinear adaptive prediction of non stationary signals. IEEE Trans. Signal Process., 43: 526-535.
10. Mandic, D.P. and J.A. Chambers, 1999. Toward an optimal PRNN-based nonlinear predictor. IEEE Trans. Neural Networks, 10: 1435-1442.
11. Hérault, J. and C. Jutten, 1994. Réseaux neuronaux et traitement du signal. Hermés, Paris.
12. Faiedh, H., Z. Gafsi, M. Mhiri and K. Besbes, 2001. Non-recurrent low-complexity neural network for the speech prediction, Advances in neural networks and applications. WSES Press, Athens, pp: 53-57.
13. Widrow, B. and S.D. Stearns, 1985. Adaptive Signal Processing. Prentice-Hall, Englwood Cliffs, NJ.
14. Rumelhart, D.E., L.L. McClelland and the PDP Research Group, 1986. Parallel distributed processing exploration in the microstructure of cognition. Vol. I, II and III. A Bradford book, MIT Press, Cambridge, MA.
15. Le, Y.C., 1987. Modèle connexionniste de l'apprentissage. PhD Thesis. Paris VI University.
16. Willams, R.J. and D. Zipser, 1989. A learning algorithm for continually running fully recurrent neural networks, Neural Comput., 1: 270-280.
17. Myers, D.J. and R.A. Hutchinson, 1989. Efficient implementation of piecewise linear activation function for digital VLSI neural networks. Electron. Lett., 25: 1662-1663.
18. Alippi, C. and G. Storti-Gajani, 1991. Simple approximation of sigmoidal functions: realistic design of digital neural networks capable of learning. Proc. IEEE Int. Symp. on Circuits and Systems, Singapore, Jun. 11-14, pp: 1505-1508.
19. Amin, H., K.M. Kurtis and B.R. Hayes-Gill, 1997. Piecewise linear approximation applied to nonlinear function of a neural network. IEE Proc. Circuits, Devices Sys., 144: 313-317.
20. Basterrextea, K., J.M. Tarela and I. Del Campo, 2002. Digital design of a sigmoid approximator for artificial neural networks. Electron. Lett., 38: 35-37.
21. Rémés, E., 1934. Sur un procédé convergent d'approximations successives pour déterminer les polynômes d'approximation. C.R. Acad. Sci. Paris, 198: 2063-2065.
22. Laurent, P.-J., 1972. Approximation et optimisation, Collection Enseignement des Sciences, 13, Hermann, Paris.

23. Muller, J.M., 1989. Arithmétiques des ordinateurs, opérateurs et fonctions élémentaires, Masson, Paris.
24. Bhatnagar, H., 2000. Advanced ASIC chip synthesis. Kluwer Academic Publishers, London.
25. Castro, J.L., C.J. Mantaset and J.M. Benitez, 2000. Neural Networks with a continuous squashing function in the output are universal approximators. Neural Networks, 13: 561-563.
26. Faiedh, H., Z. Gafsi, K. Torki and K. Besbes, 2004. Digital hardware implementation of a neural network used for classification. IEEE Proc. 16th Intl. Conf. Microelectronics, ICM'2004, Tunis, Tunisia, Dec. 6-8, pp: 551-554.
27. Gao, X.M., X.Z. Gao, J.M.A. Tanskanen and S.J. Ovaska, 1997. Power prediction in mobile communication systems using an optimal neural-network structure. IEEE Trans. Neural Networks, 8: 1446-1455.
28. Gaski, D.D.and L. Ramachandran, 1994. Introducing to high-level synthesis. IEEE Design and Test of Computers, 11: 44-54.
29. Hikawa, H., 1999. Frequency-based multilayer neural network with on chip learning and enhanced neuron characteristics. IEEE Trans. Neural Networks, 10: 545-553.
30. Narenda, K.S., 1996. Neural networks for control. IEEE Proc., 84: 1385-1406.
31. Schmid, A., Y. Leblebici and D. Mlynek, 1999. Mixed analogue-digital artificial-neural-network with on chip learning. Circuits, Devices and Systems. IEE Proc., 146: 345-349.
32. Smith, M.J.S., 1999. Application-specific Integrated Circuits. Addison-Wesley VLSI Systems Series.
33. Souani, C., 2000. Synthèse d'un système temps-réel à architecture hétérogène pour la parole et l'image. PhD Thesis, Sciences Faculty of Monastir, Tunisia.
34. Walker, R.A. and S. Chaudhuri, 1995. Introduction to the scheduling problem. IEEE Design and Test of Computers, 12: 60-69.