Buletinul Ştiinţific al Universităţii "Politehnica" din Timişoara

Seria ELECTRONICĂ şi TELECOMUNICAŢII
TRANSACTIONS on ELECTRONICS and COMMUNICATIONS

Tom 49(63), Fascicola 1, 2004

Development of advanced neural models. Software and hardware implementations

László BAKÓ[12], Iuliu Székely[3], Tihamér Sándor BRASSAI[45]

Abstract – Neurobiological researches lead to the birth of third generation (neuromorphic) artificial neural networks. One of these models is based on the natural "spiking" neural behavior, which creates the basis of our research. Following the developed mathematical "pulse reactive" model, we present our software simulator and one application of it. The FPGA built hardware spiking neuron is then introduced along with a network of these new model neurons. The modular neuron structure, acquired signals and performance analysis are given.

Keywords: Neuromorphic neural networks, Spiking neurons, Simulation, Hardware implementation, FPGA.

I. INTRODUCTION

It has been shown [3] that networks of spiking neurons are computationally more powerful then other neural models (perceptron based, with sigmoidal activation), using fewer neurons to implement the same application. We chose to develop such advanced artificial models in order to get closer to the functional behavior of natural neurons, achieving by this greater computational power using less resources and also to have a deeper understanding of their inner processes. The software simulation presented in section four has served only as an aiding tool in developing the hardware implemented neural models and networks, the main goal of our research.

II. BIOLOGICAL BACKGROUND OF THE STUDIED ARTIFICIAL NEURAL NETWORKS

A typical natural neuron has three functional components: dendrite, soma and axon (Fig.1). The dendrite is the input unit, which collects the output signals of the other neurons and transmits them to the neural cell. The cell body (soma) is the processing unit, which fulfils a complex non-linear task: if the input sum exceeds a threshold value, an output signal will be generated - an action potential [1]. This output signal is transmitted by the axon towards the receptors of other neurons. In the mammal brain a single neuron could have 10^4 postsynaptic connections, with lengths varying from µm to a few centimeters.

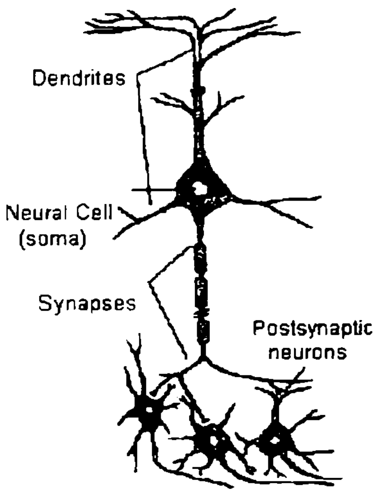

Fig. 1. The structure of a natural neuron

An action potential – measured with a intracellular electrode in the brain – consists in a short electrical pulse, of around 100 mV amplitude and 1-2 ms duration. The pulse's shape is the same during its propagation along the axon. A series of action potentials, containing regularly repeated pulses, constitute a spiking pulse row. As all spiking pulses are of similar shape, the pulse form does not contain a specific information; only the number of pulses and their time density is of major importance. The point where the axon of a presynaptic neuron is connected to the receiving dendrites of the following postsynaptic neuron, has the denomination of the synapses. In the brain most synapses are of chemical nature: the axon terminal is very close to the postsynaptic neurons and over a threshold level of the action potential the axon delivers a neurotransmitter compound, which penetrates into the postsynaptic side, causing potential level change in the postsynaptic neural cell.

[1] Teaching assistant at "SAPIENTIA" Hungarian University of Transylvania, P-ţa Trandafirilor nr. 61. Tg.-Mureş, Romania, lbako@ms.sapientia.ro
[2] PhD student at "TRANSILVANIA" University of Braşov, Romania
[3] Professor at "TRANSILVANIA" University of Brasov, Romania (IEEE member)
[4] Teaching assistant at "SAPIENTIA" Hungarian University of Transylvania, Tg.-Mureş, Romania
[5] PhD student at "TRANSILVANIA" University of Braşov, Romania

Under the influence of a spiking pulse the potential difference between the inner part of the neural cell and the environment (called membrane potential) is increasing, but if no other spikes will arrive, the membrane potential will return to the environmental level (called resting potential) (Fig. 2a)

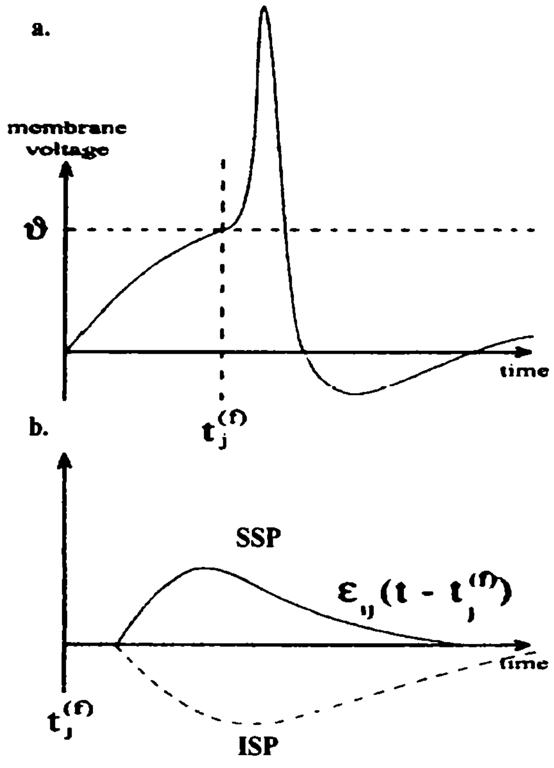

Fig. 2. a. - Membrane potential dynamics of natural neuron
b. Shape of spiking pulses (SSP - stimulating, ISP - inhibitory)

The spiking pulse could be of positive potential (Stimulating Spiking Pulse – SSP) or of negative one (Inhibitory Spiking Pulse – ISP). Assume that two presynaptic neurons (j=1,2) send SSP towards a postsynaptic neuron: j=1 neuron is firing in the moments $t_1^{(1)}$, $t_1^{(2)}$,..., $t_1^{(n)}$ and j=2 in the moments $t_2^{(1)}$, $t_2^{(2)}$,... $t_1^{(n)}$ (generally $t_j^{(f)}$). Each pulse causes a postsynaptic membrane potential increase $\varepsilon_{i1}, \varepsilon_{i2}, \ldots \varepsilon_{ij}$. If only some SSP are received by the postsynaptic neuron (i), than a near linear composition is applicable:

$$u_i(t) = \sum_j \sum_f \varepsilon_{ij}(t - t_j^{(f)}) + u_{res}, \qquad (1)$$

where u_{res} is the residual, resting membrane potential. If there a greater number of SSP are received, the summing law of pulses is no longer linear. If the potential u_i reaches a critical threshold level ϑ, the postsynaptic neuron will deliver an own spiking pulse, which will be transmitted through the axon of the neuron i towards the synapses of further neurons. After delivering the firing pulse the membrane potential of the neuron i will drop back to the stand-by level. Practically, in biological neurons, around 20-50 presynaptic spiking pulses must be received in a short time, to exceed the threshold level.

III. MODELING OF SPIKING NEURAL NETWORKS

In a spiking neural network (SNN) we define a finite set of firing neurons (FN) and a set $E \subseteq VxV$ of synapses. To each $i \in V$ neuron belongs a threshold function $\vartheta_i : R^+ \to R^+$ and to each $\langle i, j \rangle \in E$ synapses belongs an answer function $\varepsilon_{ij} : R^+ \to R$ [3],[4] (Fig. 2b).

This answer function could be a positive one, when it is a stimulating spiking pulse – SSP (Fig. 2b$_{SSP}$) or a negative one, when it is an inhibitory spiking pulse – ISP (Fig. 2b$_{ISP}$).

When a neuron j is firing, the time characteristic could be expressed by: $w_{ij}.\varepsilon_{ij}(t - t_j^{(f)})$, where $\varepsilon_{ij}(t - t_j^{(f)}) = 0$ if $t - t_j^{(f)} < \Delta_{ij}$, Δ_{ij} being the transmission time of the pulse from the FN to the postsynaptic receiving neuron; w_{ij} is a weighting factor, which is positive for SSP and negative for ISP.

If $t - t_j^{(f)} \geq \Delta_{ij}$, the answer function has the shape presented in Fig. 4.

The pulse reactive model (PRM) is, may be, the most appropriate method to model SNN, the task being to evaluate the firing moments of FN i ($t_i^{(1)}$, $t_i^{(2)}$,...$t_i^{(n)}$) as a function of the firing moments of j presynaptic FNs ($t_j^{(1)}$, $t_j^{(2)}$,... $t_j^{(n)}$).

The membrane potential of a cell (soma) of the i-th neuron is

$$u_i(t) = \sum_{i,j} w_{ij}.\varepsilon_{ij}(t - t_j^{(f)}), \qquad (2)$$

which expresses the contribution of j presynaptic FNs, generated before the moment t. The PRM handles also with a damping factor $\eta(t - \hat{t}_i)$, which is a function of $\hat{t}_i$ -the moment of the generation of the last spiking pulse of FN i, before the time t. If $t - \hat{t}_i < 0$ or $t - \hat{t}_i > 0$ and high enough, the damping factor is zero. If $t - \hat{t}_i > 0$, but of small value (that means the i FN just has generated a spiking pulse), the damping factor has strong negative value and inhibits the FN i to generate a new spiking pulse.

Comparing to the Hodkin-Huxley model [2], [5], the pulse reactive model (PRM) is more suitable to computer analysis, as there are no differential equations are handled in the model. If we choose appropriate values for $\eta, \vartheta, \varepsilon$ functions, the PRM could model quite good the dynamics of the natural, biological neurons.

In our model we used a coding algorithm, where each spiking pulse has a separate importance, similar to the bits in the computers. As the time is continuous, a single spiking pulse could include more information than a single bit, as the arriving time t could encode the analog value $(t-T)$, when T is a reference time value. In this respect a FN acts as a coincidence detector, i.e. will generate a spiking pulse only if a great enough number of SSPs will arrive to the neuron cell (soma), almost simultaneously.

If all the transmission delays Δ_{ij} between the FN i and the presynaptic j FNs are identical, then the FN i, which has a high threshold value ϑ_i, will fire only if all presynaptic j FNs will have the firing moments $t_j^{(f)}$ very close to each other (practically are simultaneous). If the Δ_{ij} are different for several j p[illegible] presynaptic neurons will comply to a certain scheduling in firing times: i will fire if for any j, $t_j \approx T-\Delta_{ij}$ [6].

IV. THE DEVELOPED SIMULATION ENVIRONMENT AND THE TEST RESULTS

A. *The simulation algorithm*

The main aim of the simulation procedures is the computation of the networks parameters, which are necessary to generate spiking pulses in a given moment of the network simulation. The spiking pulses are generated from the membrane potential of neurons and with the threshold function ϑ_i. Consequently it must process the pre- and postsynaptic pulses to evaluate them as membrane potential effect for a coming neuron, which will fire in the next time-step. In the same time the synaptic weightings and delays must compute.

In the FNN model there the classical perceptron model matrix vector algorithms, used in the previous generation artificial networks, are not applicable. Here the neurons and synapses need separate computations, which are possible in sequential data processing. In this respect every time step of the simulation must be divided in two phases: in the first phase we compute the spiking pulses of all the neurons and after that we use them in the second phase, when we proceed the necessary operations to transmit the spiking pulses. Based on this two-phase evaluation, the next algorithm is set up (Fig. 3):

- In the first phase we compute the values of the membrane potentials and threshold values, based on the data read from the Neuron Memory. If a membrane potential reaches a value that exceeds the threshold, then spiking pulse (SP) will be generated, which is saved in a spiking pulse SP List, together with the identificator of the FN.
- In the second phase the values read from the SP List are weighted (with coefficients w_{ij}) and using the Neural Network Topology the SP is transmitted towards the postsynaptic neurons. At this phase we could use the learning algorithm, modifying the effectiveness and timing of the synapses.

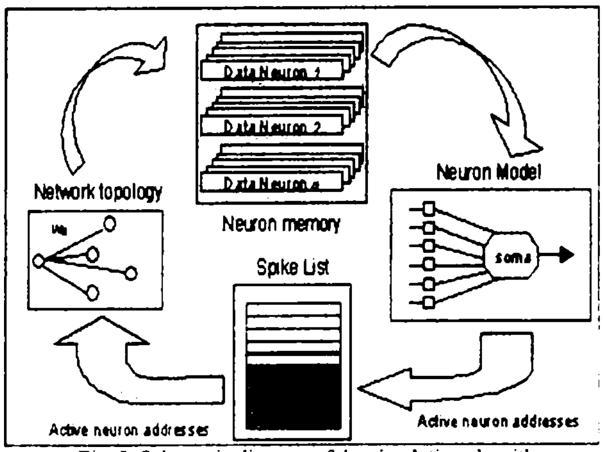

Fig. 3. Schematic diagram of the simulation algorithm

After these phases the state of the network is deterministic and the next time step of the computation may start.

As learning algorithm we used the Hebb learning algorithm. This consists in a set of learning rules, which allow the tuning of the synapses based on the pre- and postsynaptic neurons activity.

Generally speaking, the Hebb learning has the following four components:

1. A neural connection is fortified (the synaptic efficacy is increased) if the pre- and postsynaptic neurons are active simultaneously (are firing).
2. A neural connection is weakened (the synaptic efficacy is decreased) if only the presynaptic neuron is active.
3. A neural connection is weakened (the synaptic efficacy is decreased) if only the postsynaptic neuron is active.
4. The neural connection is unchanged if none of the neurons is active.

B. *The implemented software simulation*

The complexity of the simulation task determined a very careful selection of the development tool and environment. As it was presented in the previous sections the studied natural neural networks as well as the developed artificial ones relay on signal timings, therefore timing has a very important role. Hence, we had to find a solution which gives tools for exact and schedulable timings beside the proper programming environment. After running a few tests and considering their results we decided to drop the path of an interpreter (Matlab) or a compiler (Visual C++) under the Windows operating system. Instead we turned towards a C++ compiler running on a RedHat Linux operating system. This way we have gained not only much better timing tools but relatively faster 2D and 3D plotting interfaces (Gnuplot, Povray) and a modern and useful way of saving network structures with the XML file format.

A considerable amount of our software developing work was based on the previously developed spike-response model and threshold-based learning. Here we can briefly mention the programming of the

behavior of the individual spiking neurons and their synapses, and of the layer building and connection establishing procedures. Any 3D network can be built either giving the architecture neuron by neuron or layer by layer.
The networks can possess feedbacks, for instance one can create a recurrent network formed of several layers where the outputs of the last layer are connected to the inputs of the second layer (the first layer neurons are special neurons without synapses, accepting only one input), thus creating a pulsing output. This can be used as some form of synchronization signal for other groups of neurons.
The simulation environment, therefore uses three types of neurons as follows: input layer neurons, hidden layer neurons and output layer neurons.

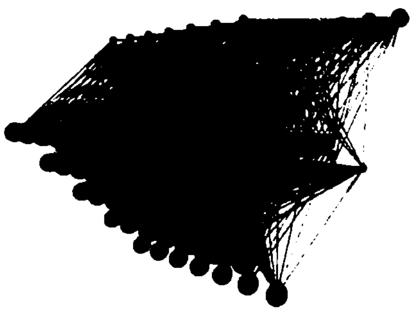

Fig. 4. The tested spiking neural network built in the simulation environment

The test application we have prepared is a letter recognition from a 7x5 matrix of inputs. The developed spiking neural network architecture is presented in a spatial form in Fig. 4, where the matrix-shape arranged blue balls represent the input neurons, the smaller inner balls the hidden layer neurons and the ones in the background the neurons of the output layer. The meeting points of the connecting lines between neurons are the synapses. The hidden layer has the same number of neurons as the input layer, while the output layer has seven neurons. As the output neurons have binary outputs these may stand for the elements of a seven segment display. Though, an additional binary logic is necessary to display the characters correctly on the seven segment display. This occurs because the network - with the currently implemented unsupervised learning algorithm - performs only a categorization of the noised matrixes given at the input.
The input matrixes are the representations of different characters presented randomly to the network with adjustable noise percentage. The simulation is composed of a prescribed number of time-steps. At each time-step a different input is presented to the network and all the necessary computations – for all neurons and synapses – are executed.
If enough time-steps have elapsed the output of the spiking neural network will start to present similar values for a certain input matrix, hence it will learn the given task. In order to be able to follow the evolution of the system at each time-step all active (spiking) neurons are marked and saved.

V. FPGA IMPLEMENTATION OF THE SPIKING NEURONAL MODEL

An artificial neural network (ANN) is a parallel and distributed network of simple nonlinear processing units interconnected in a layered arrangement. *Parallelism, modularity* and *dynamic adaptation* are three computational characteristics typically associated with ANNs. FPGA-based reconfigurable computing architectures are well suited to implement ANNs as one can exploit concurrency and rapidly reconfigure to adapt the weights and topologies of an ANN.
FPGA realisation of ANNs with a large number of neurons is still a challenging task because ANN algorithms are usualy "multiplication-rich" and it is relatively expensive to implement multipliers on fine-grained FPGAs. By utilizing FPGA reconfigurability, there are strategies to implement ANNs on FPGAs cheaply and efficiently.
One of these strategies, which we have applied, consists in implementing third generation neural models on FPGAs, because it is possible, as you can read further in this paper, to build them without using any multiplicator circuits. Most multiplier-like task are fulfilled by sequential counter logic circuits. This makes our approach more cost-efficient than the hardware built, perceptron based ANNs. Due to the high level of paralellism achieved in this manner speed enhancement is also obvious, as Section VI and Section VII present.
Based on the mathematical model and the results of the simulated network we developed a hardware implementation of spiking neural networks. The two main parts of the neuron model, the synapse and the soma, were designed and developed separately. As the previously section of this paper presented, one of the developed neural model's specific properties is that the inputs and outputs can be easily given as binary values (pulses). This was one of the reason which led to the digital hardware implementation. The device used was a XESS XSA100 prototyping board having as main module a XILINX XC2S100 Spartan II FPGA (100k logic gates). Beside the FPGA chip there is a CPLD IC for the PC parallel port connection interface, a 16Mb SDRAM, a 256Kb FlashRAM memory module present on the prototyping board, the FPGA being driven by a programmable oscillator with the maximum frequency of 100 Mhz.

A. *The FPGA implemented synapse*

Considering the behavior of a cerebral synapse the main function of the circuit developed by us as the artificial synapse is a pulse multiplier, although it doesn't contain any real multiplicators.
As the architectural schematic in Fig. 5 presents, the synapse is also made up by two – structurally and functionally – separate units.

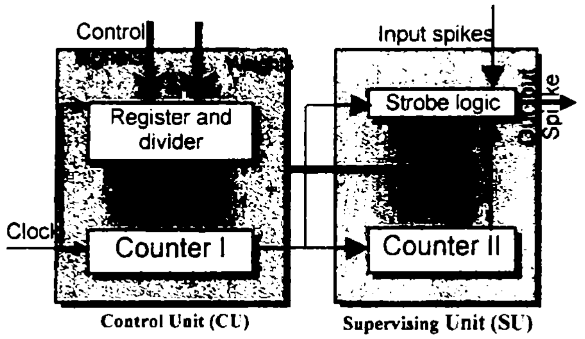

Fig.5. The synapse

The Control Unit (CU) consists of a register and a counter and performs the following tasks on the rising edge of the clock signal:

- If the Write (WR) signal is active, it updates and stores the weight value of the synapse from the external weight bus;
- If the Read (RD) signal is active, the CU writes out the weight value to the external weight bus;
- Signals PRS (pre-synaptic spike) and POS (post-synaptic spike, axonal spike) are used to implement the learning algorithm which described in ***VHDL code*** looks like this:
  if PRS='0' and POS='0' then new_weight:=old_weight; end if;
  if PRS='1' and POS='0' then new_weight:=old_weight-1; end if;
  if PRS='0' and POS='1' then new_weight:=old_weight-1; end if;
  if PRS='1' and POS='1' then new_weight:=old_weight+1; end if;
- Counter I (limited to four bits, due to lack of space on the FPGA) is increasing or decreasing as the actual weight is modified, value which is then transmitted to the Supervising Unit (SU).

The SU watches for an incoming spike pulse. When such a pulse arrives the SU emits a number of output pulses, exactly as much as the weight value shows. When Counter II reaches this value, the Validation Logic inhibits further output spikes.

These output spikes are then transmitted towards the neuron body, the soma for further processing.

B. *The FPGA implemented soma*

The soma module can be considered as the Central Processing Unit of neuron model. It is the most complex part of the whole neuron, hence it uses the most circuit logic blocks (CLB's) of the FPGA. Consequently, it has required the development of various version until a fairly optimized configuration has been reached. Though, it is still too large, uses too much of the FPGA logic, to allow large networks to be built.

The architecture of the soma is built on four modules (Fig. 6). The main task of the soma is to calculate the membrane potential using the input spikes from the synapses, compare it to the threshold potential and if

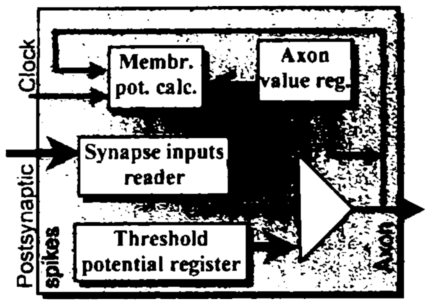

Fig. 6. The hardware implemented soma

the latter is the smaller one it has to emit an axonal spike. On the falling edge of the clock signal, the SYNIN unit of the soma reads the output values of the synapses and performs a multiplication with a programmed factor, then sums these into an input value. Through an internal bus, this value reaches the MPCU unit where it is added to the previous value of the membrane potential, stored here on seven or eight bits. If there were no input spikes, then the membrane potential is gradually decreased, down to the resting potential (4~5% of the maximal membrane potential value) to reflect the natural neuronal behavior presented in section 2. The MPCU module also contains a time-frame creation counter used to limit the time while a certain number of input spikes have to arrive in order to cause an axonal spike. Here, it has to be noted, that we implemented the soma only for excitatory synapses.

The updated membrane potential is then compared to the adjustable threshold-potential. If the required criteria is met (membrane potential greater then threshold-potential), an axonal spike is emitted and the neuron is placed into hyperpolarisational phase, when the membrane potential is set below the resting potential (zero).

VI. THE FPGA BUILT NEURON

After months of extensive experimental testing on single neuron models we have selected one with nine synapses to be used in a multiple neuron test schematic (a test neural network). The mentioned model represents the membrane and the threshold potential on seven bits, with four bit weight values while the spike emitting time window is set to 16 clock cycles. As first test application the classic problem of classifying two different shapes was selected. One of the great assets of these neural models is the fact, which we have show in the next section, that the mentioned problem can be solved with a network consisting of only two spiking neurons. The network (Fig. 8) is able to learn to distinguish between any two shapes represented is 3x3 matrix of points, eliminating the effect of noise.

Communication between a PC and the XSA prototyping board is possible through the parallel port Interface (8+4 bits).

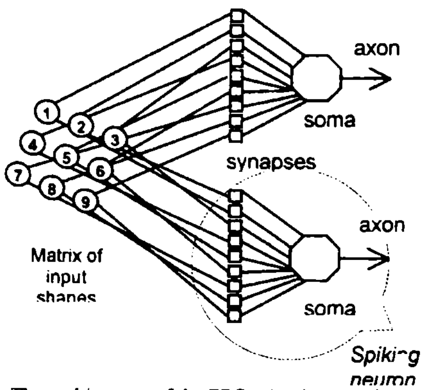

Fig. 7. The architecture of the FPGA implemented test neural network

I- ---- t- b- -b' 'o d 'iver '' e inpu' s imulus for he network from a PC (using a C++ program) we used serial data input by implementing a special (serial to parallel) interface on the FPGA. This Serial Input Device (SID) assures the addressing and loading of initial wei ht values into the s napses, followed by the input spikes corresponding to the different shapes (T and H or + and X). The SID uses separate clock signal and several control signals sent through the p_ _l_.l ,___ ____ . .gis.er. Th w ._.t valu s and .... axon signals can be read back alternatively to the PC using the parallel port status register.

A. *Measurement results on the test neural network*

In the initial experimental measurements the values of the synapses during learning were read back from each neuron after those were presented with both input shapes. This slow communication through the SID allowed accurate measurements but was not working real-time. To achieve this, we switched the input feeding from the parallel port to a PIC16F876 microcontroller working at 20 MHz, connected to the XSA board (all 9 input bits connected parallely). Hence, we could present to the SNN a new input shape at every two clock cycles of the PIC (2x50=100ns).

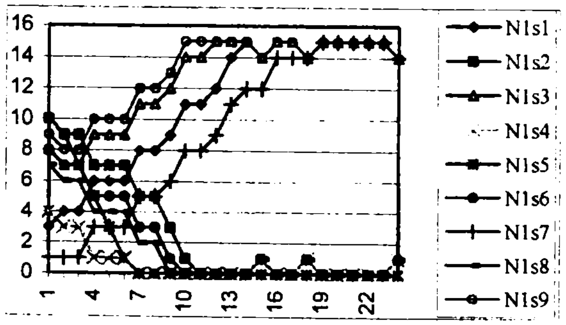

Fig. 8. Learning: weight values of one of the neurons

Fig. 8 shows how one neuron learns to identify one of the input shapes. One can observe, that the weight values (Y axis) stabilize around the extremes (0 & 15) depending on which input received incoming spikes. This process takes around 20 time slices (X axis) each time slice taking 16 clock cycles. At the PIC's frequency this yields: 16x20x100ns=*32μs learning time.* In other w rds, the network learns t identify two shapes in ~32μs. If we switch off the learning and connect the clock to 100MHz (XSA board maximum), then the SNN will recognize the learned shapes in ~3,2μs. This means a big improvement in speed compared to the previously presented software enviro..me..t, wh... a simila. ..tw..k l.arn.d th. same task in about ~200ms or to a perceptron network in Matlab with it's ~300ms. Fig. 9 and 10 presents data acquired with a 34 channel Tektronix Logic Analyzer from the working neural network impl.m.nt.d on th. FPGA.

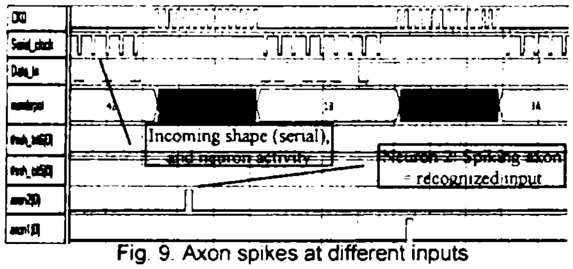

Fig. 9. Axon spikes at different inputs

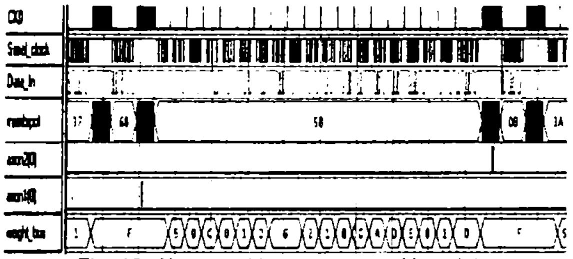

Fig. 10. Alternated input shapes with weight

VII. CONCLUSIONS

So far we have a working, hardware implemented spiking neural model capable of classification through unsupervised learning. The main drawback at the moment is the size of one neuron (~15000 logic gates) which undoubtedly needs optimization. We also plan to develop more complex learning algorithms by tuning several neural parameters. The ultimate goal is to build a system with structural self-organizing ----b'''-- f-l'-w-d b- ---'---i-- -- --d---ri l adaptive control processes.

The current research is part of the prime author's PhD thesis theme which is under development since 2003 and has been partly funded by the Research Institute of the Sapientia Foundation.

REFERENCES

[1] W. Gerstner and W.M. Kistler, "Spiking Neuron Models. Single Neurons, Populations, Plasticity"; *Cambridge University Press*, 2002.

[2] T.P. Trappenberg; "Fundamentals of Computational N-------i-----"; *Oxf--d Univ r P* , 2002.

[3] W. Maas, "Networks of spiking neurons: the third generation of neural network models". *Neural Networks*, 10(9):1659-1671, 1997.

[4] W. Maas, "A simple model for neural computation with firing rates and firing correlations", *Network: Computation in Neural Systems*, 9:1-17, 1998.

[5] W. Maas and C.M. Bishop, "Pulsed Neural networks", *MIT Press*, 1999.

[6] J. J. Hopfield, "Pattern recognition computation using action potential timing for stimulus representation". *Nature*, 376:33-36, 1995.