A Neuron Model for FPGA Spiking Neuronal Network Implementation

Liviu ŢIGĂERU, Gabriel BONTEANU
Gheorghe Asachi Technical University of Iasi, 700506, Romania
ltigaeru@etti.tuiasi.ro, gbonteanu@etti.tuiasi.ro

***Abstract*—We propose a neuron model, able to reproduce the basic elements of the neuronal dynamics, optimized for digital implementation of Spiking Neural Networks. Its architecture is structured in two major blocks, a datapath and a control unit. The datapath consists of a membrane potential circuit, which emulates the neuronal dynamics at the soma level, and a synaptic circuit used to update the synaptic weight according to the spike timing dependent plasticity (STDP) mechanism. The proposed model is implemented into a Cyclone II-Altera FPGA device. Our results indicate the neuron model can be used to build up 1K Spiking Neural Networks on reconfigurable logic suport, to explore various network topologies.**

***Index Terms*—spiking neural network, neuromorphics, biological system modeling, field programmable gate arrays, very large scale integration.**

I. INTRODUCTION

Biological neurons are complex structures that communicate each other by means of 1ms voltage pulses, called spikes. Each neuron transforms the spatio-temporal patterns applied to presynaptic inputs into output spike trains that are distributed along the axons to other neurons, where they evoke postsynaptic responses. If a postsynaptic neuron receives spikes from several presynaptic neurons within a narrow time window, its membrane potential increases and may reach a critical threshold, firing a postsynaptic spike [1].

The Spiking Neural Networks (SNNs) are a relatively recent class of artificial neural networks [2], more biologically plausible than the conventional rate-coded counterpart, built up with complex computational units that mimics the biological neuron dynamics. Consequently, the SNNs can generate and reproduce behaviors similar to biological neural systems, being appropriate in experiments to model the operational functionality of the brain and engineering applications requiring the brain peculiar abilities, respectively.

Cortical neurons show a wide variety of soma and synaptic responses to their synaptic input signals. The neuroscientists have developed various neuron models that vary in complexity between very detailed and computational expensive models to simpler and computational effective models. Among these, the most complex model, capable to reproduce the rich neuronal dynamics is the one developed by Hodgking–Huxley in the 50th [3-7]. Despite its impressive repertoire of spiking behavior, the Hodgking–Huxley model is described by a set of complex nonlinear differential equations that make it inappropriate for silicon implementations.

A good compromise between computational cost and repertoire of spiking is the neuron model proposed by Izhikevich [8]. The neuron model is described by means of a two differential equations system, describing the evolution of a membrane potential variable and a slow variable, together with a reset mechanism.

One of the most popular neuron model is the integrate-&-fire (I&F) neuron model [9]. The simplicity of the I&F neuron model allows to implement large, massively parallel network of spiking neurons, where de computational expenditure is a challenging design issue, at the cost of a rough repertoire of spiking behavior.

In the recent years much research has been devoted to the silicon implementation of biologically inspired SNNs. Attractive analogue circuits where reported for Hodgking–Huxley [10], Izhikevich [11] and I&F [12] neuron model implementations. The analog approach provides compact solutions, which lead to low silicon area and electrical power consumption, allowing silicon implementations for large parallel neuronal architectures that speed up the information processing. However, the analogue solutions have some important inconveniences. The most important ones consist of the poor flexibility, the lack of effective analogue circuits for storing elements and low accuracy. The analogue solutions are usually applied to silicon implementations of application specific SNNs architectures, but are inappropriate as hardware support for applications that intend the neural network topologies exploration.

Digital technology solve the drawbacks of the analogue circuits and may serves as an effective solution for reconfigurable logic platform implementation, appropriate for neural network topologies exploration. Digital circuits were reported for I&F [13] and Izhikevich [14] neuron model implementations. Because of the main goal of the scientists is to build up powerful computational platform based on large SNNs, one of the major concerns is the logic resource expenditure required for hardware implementations.

In the present paper, we propose a neuron circuit that captures the basic elements of the biological neuron dynamics, suitable for digital implementations of reconfigurable SNNs. The neuron model consists of a soma circuit, that reproduces the neuronal membrane potential dynamics and a synaptic circuit that implements the learning rule, based on the spike timing dependent plasticity (STDP) mechanism [15], a class of spike driven learning rule, well suited for VLSI implementation.

We describe in detail the mathematical model of the

This work was supported by the CNCSIS-UEFISCU Romania, under Grants PN II-IDEI 740/2008 and PN II-IDEI 1552/2008.

Digital Object Identifier 10.4316/AECE.2011.04005

proposed neuron model in section II and its circuit architecture development in the section III, respectively. Section IV-V presents the implementation of the neuron circuit into a Cyclone II-Altera FPGA device and simulation results respectively. Finally, some concluding remarks are drawn in the section VI.

II. The Neuron Mathematical Model

The proposed neuron model reproduces the basic behavior of the biological neuron, observed at the soma level and synaptic inputs respectively. The soma dynamics behave as follows: in the absence of the presynaptic spikes, the neuron is in the resting state (RST state), during its membrane potential has a constant value called the resting potential. The presence of the presynaptic spikes at the synaptic terminals of the neuron leads to its depolarization (DEP state). During this state, the membrane potential increases directly proportional to the presynaptic activity, recorded at the synaptic input terminals. The cease of the presynaptic activity leads to the return of the membrane potential back to the resting potential, the neuron keep waiting in the resting state, to receive new presynaptic spikes. On the other hand, if the presynaptic activity is intense, the depolarization of the neuron can become sufficiently large to reach a critical threshold, causing the neuron to fire a postsynaptic spike. A postsynaptic spike firing is followed by a hyperpolarization phase, during the membrane potential goes under the resting potential. The hyperpolarization starts with an absolute refractory period (ARP state), during the membrane potential is kept to a minimum value, the neuron being completely insensitive to any presynaptic activity. The hyperpolarization follows with a relative refractory period (RRP state), during the membrane potential slowly returns back to the resting potential, the neuron being less sensitive to the presynaptic activity, compared to the depolarization state.

The membrane potential dynamics is described by a set of four elementary equations:

$$v_M(t) = v_M(t-1) + k + \sum_{i=0}^{N-1} \alpha_i \cdot w_i(t-1) \cdot S_i(t-1) \quad (1)$$

$$\text{if } v_M(t) = V_T \text{ then } S_j(t) \to 1 \text{ and } v_M(t) = V_{MIN} \quad (2)$$

$$k = \begin{cases} -k_D & DEP \\ 0 & ARP \\ k_R & RRP \end{cases} \quad (3)$$

$$\alpha_i = \begin{cases} 1 & DEP \\ 0 & ARP \\ <1 & RRP \end{cases} \quad (4)$$

where:

v_M = the membrane potential,

S_i = the presynaptic spike,

w_i = the synaptic weight,

N = the number of the synaptic inputs,

t = the current iteration,

V_T = the firing threshold,

V_{MIN} = the minimum value of the membrane potential,

k = the change factor of the membrane potential in the absence of the presynaptic activity,

α_i = the sensitivity coefficient of the neuron, that modulates the synaptic activity depending on the neuron state,

S_j = postsynaptic spike.

The synaptic behavior is modeled by means of a multiplicative operator, applied to the presynaptic spike and the synaptic weight, which is updated according to the equation,

$$w_i(t) = w_i(t-1) + \Delta w_i(t) \quad (5)$$

where Δw_i represents the change of the synaptic weight.

The proposed neuron model uses the STDP mechanism as learning rule, a large used method for learning implementation in SNNs, based on the Hebb rule of the synaptic efficacy adaptation [16]. According to this learning rule, the variation of the synaptic efficacy (synaptic weight) is modified according to the time interval between the moment that a postsynaptic spike is fired, denoted t_{POST}, and to the moment that a presynaptic spike is detected at the synaptic terminal, denoted t_{PRE}, as is shown in Fig.1.

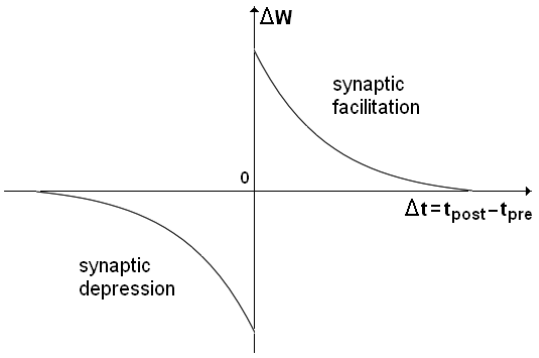

Figure 1. The STDP mechanism.

If a presynaptic spike arrives at the synaptic input, within a critical time window called learning window, before a postsynaptic spike is emitted, it is considered that the postsynaptic spike was fired because of the presynaptic activity. In this case, the synaptic efficacy is facilitated. Conversely, if the postsynaptic spike is emitted soon before the presynaptic spike arrives, it is considered that the presynaptic activity has not any effect on the postsynaptic neuron activity. In this case, the synaptic efficacy is depressed. In the proposed model, the STDP mechanism is introduced by means of the equation,

$$\Delta w_i(t) = \tau_R \cdot w_{ij} - \tau_F \cdot \left(LW - w_{ij}\right) \quad (6)$$

which describes the updating rule of the synaptic weight. In (6) τ_R is the rising time of the synaptic efficacy and τ_F is the falling time of the synaptic efficacy, respectively. The LW parameter is the learning window and w_{ij} is a variable that depends on the number of iterations elapsed between the arrival moment of a presynaptic spike at a synaptic terminal of a neuron, and the firing moment of the postsynaptic spike at the same neuron. If the postsynaptic spike is fired on the next iteration after the presynaptic spike detection, the w_{ij} variable has a maximum value. Otherwise, w_{ij} value decreases in time, in proportion as the number of iterations counted between the moment of the presynaptic spike detection and the moment of the postsynaptic spike firing, increases. When w_{ij} reaches 0 value, the learning window has finished and the w_{ij} value remains locked at this value until a new presynaptic spike is detected.

III. The Neuron Architecture

A. The Optimization Procedure

The calculation of the equation (1) involves 2×N multiplications and N+1 additions, requiring a lot of logic resources for hardware implementation. Because of the neuron model is intended to be used as a computational unit for large SNNs, it is a demanding design issue to optimize it for an effective silicon implementation.

The equation (1) may be sequentially computed in N+1 clock cycles, by means of an adder-accumulator structure. In addition, the accumulator may be used to cancel the sum (1), during the absolute refractory period (ARP state). The multiplications within the terms of the sum $\Sigma\alpha_i w_i S_i$ in the equation (1) may be completely elliminated by conveniently choosing the sensitivity coefficients values and turning to advantage the fact that any spike takes only two binary values {0,1}. Thus, because of the membrane potential dynamics is solely handled by the accumulator during the ARP state, the α_i values does not need to be explicitly specified in this stage. According to this, α_i may be defined as:

$$\alpha_i = \begin{cases} 1 & DEP \\ \dfrac{1}{2} & RRP \end{cases} \quad (7)$$

Consequently, the multiplications of synaptic weights w_i with the sensitivity coefficients α_i may be reduced to the choice between two synaptic values, w_i and $w_i/2$ respectively, according to the neuron state. This procedure may be described by the equation,

$$1 \cdot w_i(t-1) \cdot neuron\_state + \frac{1}{2} \cdot w_i(t-1) \cdot \overline{neuron\_state} \quad (8)$$

where the "neuron_state" signal control delivers the neuron state, being '0' logic during the RRP state, and '1' logic during the DEP state. The division by 2 of the synaptic weight w_i may be performed by a 1 bit right shift operation of the w_i value. This procedure does not necessary requires a distinct shift register and may be simplified by implementing it at the layout level, by the wire concatenation between the layout ground ('0' logic) with the [msb ,…, lsb+1] w_i field, the new w_i value resulting as [0, msb ,.., lsb+1].

Similarly, because of a spike takes only binary values, the multiplication of the presynaptic spikes S_i with the $\alpha_i \cdot w_i$ terms may be reduced to the choice between '0' logic and $\alpha_i \cdot w_i$ value, according of the presynaptic spike value. Thus, any term of the sum $\Sigma\alpha_i \cdot w_i \cdot S_i$ in the equation (1) could be obtained by a procedure described as:

$$\alpha_i \cdot w_i(t-1) \cdot S_i + 0 \cdot \overline{S_i} \quad (9)$$

Both relations, (8) and (9) respectively, match the generic equation,

$$y = do \cdot \overline{sel} + d1 \cdot sel \quad (10)$$

which represents the operating equation of a two data input multiplexer. Consequently, two small 2 data input multiplexers could replace the multipliers involved to get any term of the sum $\Sigma\alpha_i \cdot w_i \cdot S_i$ in the equation (1), saving a significant quantity of logic resource that otherwise would be required to implement the multipliers.

Moreover, the comparator required to implement (2) may be removed by an appropriate choice of the critical threshold V_T value. If the membrane potential value is represented by means of B bits and the critical threshold value is chosen as $V_T=2^B-1$, the condition (2) becomes equivalent with an overflow condition detected at the carry output signal of the adder, used to implement the equation (1).

Each synaptic weight update process requires two multipliers, an adder and two subtractor circuits respectively, rising the quantity of logic resources required to implement the synaptic circuit. This approach leads to a critical design issue when the neuron holds many synaptic inputs. But, if the terms required to compute each w_i value are sequentially applied to the inputs, because of the equation (1) is computed in a sequential manner, the synaptic weight computation procedure may be implemented by a single set of the above mentioned computational circuits. Subsequently, each updated synaptic weight value is provided to the circuit that implements the sum (1) in a pipelined fashion, as is suggested in Fig. 2, where T_C is the computation time period, and T_t is the iteration time period. According to this approach, each updated synaptic weight w_i value is fed one cycle before the term $\alpha_i \cdot w_i \cdot S_i$ is actually used in calculations.

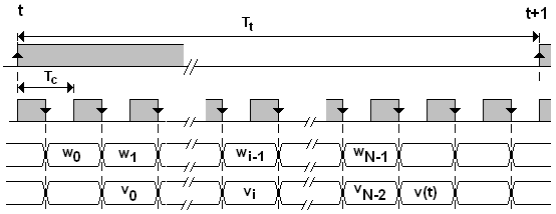

Figure 2. The pipeline technique for membrane potential calculation.

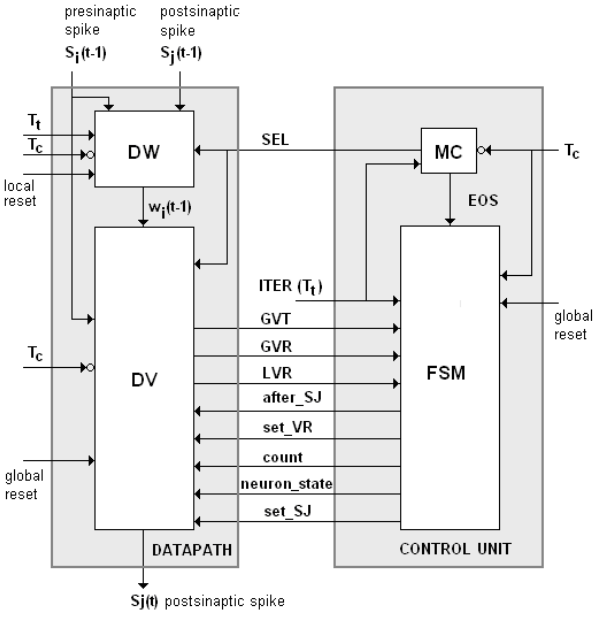

Figure 3. The neuron architecture circuit.

The multiplication with the rising/falling time constants may be avoided if both τ values are chosen equal to unity. Furthermore, if a peculiar representation of the w_{ij} is used, a single adder/subtractor circuit may replace both adder and subtractor circuits. This solution is detailed, in section III.B.

Finally, it can be concluded that, based on the optimization techniques described above, each neuron requires an adder accumulator structure for the membrane

potential computation, an adder/subtractor circuit, N registers for storing synaptic weight values and a set of multiplexers to guide the operands involved in the current calculations. The identified logic resources set lead us to a conventional computational architecture, which consists of a datapath and a control unit.

The circuit architecture of the neuron model is depicted in Fig. 3. The datapath unit executes the operation required to implement the neuron dynamics. The datapath unit consists of a synaptic weight and membrane potential computation circuits, that model the synaptic inputs and the soma dynamics of the neuron, denoted DW and DV respectively. The control unit receives information about the datapath state and generates control signals to synchronize the datapath operation cycles. The control unit consists of a master counter, denoted MC, and a Moore finite state machine, denoted FSM. In the sequel, the description of these circuits is provided in detail.

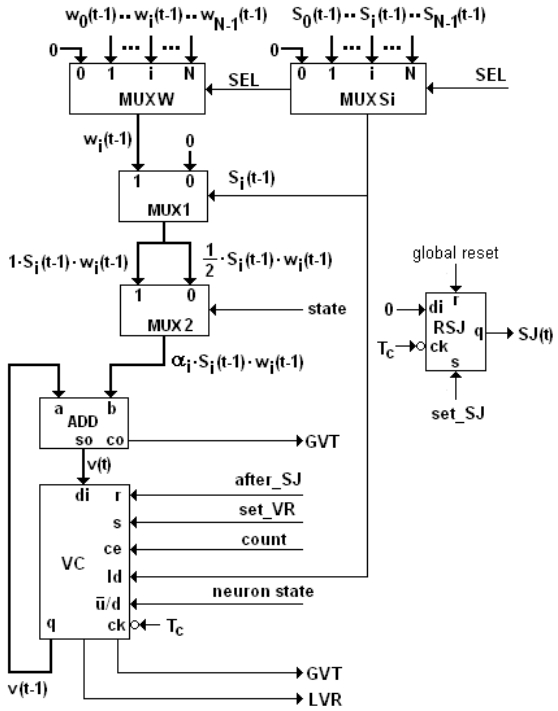

Figure 4. The soma circuit.

B. The Datapath Unit

The structure of the soma circuit is depicted in Fig. 4. We have chosen an 8 bits resolution for the membrane potential value, with V_T=255, V_R=63 and V_{MIN}=0, where V_R is the resting potential. However, the structure is scalable and a greater resolution may be adopted, if the application requires.

The circuit computes the sum (1) in a sequential manner. Each T_C clock cycle, the input multiplexers MUXW and MUXSi take on a single synaptic weight - presynaptic spike pair, depending on the SEL selection signal, generated in the unit control by the MC master counter. Because of the SEL signal also controls the currently computed synaptic weight in the synaptic circuit, the w_i value is computed one T_C cycle before the respective term to be used in the soma circuit to compute the membrane potential value, as can be seen in Fig. 2. Thus, in order to synchronize the moments when the synaptic weights values are delivered to the soma circuit with the moment when the presynaptic spikes arrive to the synaptic inputs, it is required to delay one T_C period the moment when each term $\alpha_i \cdot w_i \cdot S_i$ is computed relative to the moment when the synaptic weight w_i is computed. The required delay is accomplished if both the synaptic weight w_i and the presynaptic spike S_i are connected beginning from the second data input of the input MUXs, denoted as 1 in the Fig. 4, leaving the first data input (denoted 0) to the ground.

Each term of the sum $\Sigma\alpha_i \cdot w_i \cdot S_i$ is computed at MUX1-MUX2 level, according to the procedure described in the optimization section. The sum $\Sigma\alpha_i \cdot w_i \cdot S_i$ is computed by means of an add-and-accumulate technique, by the ADD adder and the VC reversible counter.

The reversible counter with loading facilities was preferred as accumulator element. This solution is more effective, allowing a natural implementation of the v(t-1)+k partial sum (with k's values used as increment and decrement steps), removing the adder/subtractor circuit. In addition, this solution simplifies the logic control of the control unit.

The counter operation depends on the neuron state during the current iteration, and is controlled by means of the control signals provided by the control unit. Thus, the counter is synchronously initialized to the V_{MIN} value on the negative edge of the T_C clock signal, after a postsynaptic spike is fired (when "after_SJ" control signal is asserted). It is synchronously forced to the resting potential V_R on the negative edge of the T_C clock signal, whenever the neuron enters in the resting state (when set_VR signal is asserted). Otherwise, on the first T_C clock cycle after the current iteration begins (after the positive edge of the T_t signal is arrived), while the "count" signal is asserted, the counter is decremented with a k_D=8 step, or incremented with a k_R=4 step, depending on the value of the "neuron_state" signal. Subsequently, after the first T_C clock cycle ends, the counter waits to load the adder output value, depending on the presynaptic activity.

The membrane potential is permanently monitored by the control unit via LVR and GVT lines, asserted when $v \le V_R$ and $v \ge V_T$, respectively. As $v \ge V_T$ (V_T=255, the maximum 8 bits value), an overflow condition occurres at the adder and the soma circuit send a firing condition to the control unit via GVT line.

The RSJ register is used to trigger the postsynaptic spike. It delivers '1' logic as long as "set_SJ" control signal is asserted, otherwise, its output is kept to '0' logic.

The structure of the synaptic circuit is depicted in the Fig. 5. This circuit computes the current synaptic weight according to a learning rule.

The multiplexers – demultiplexer group provides the operands required in the current calculation of the synaptic weight and the w_i values for the storing elements, respectively, depending on the current value of the selection signal SEL.

The down counter CWi, i=0÷N-1, counts the iteration number elapsed between a presynaptic spike detection moment and a postsynaptic spike firing moment occured at the same neuron. A presynaptic spike detection reset the CWi counter to zero, synchronously to the positive edge of the T_t clock signal, which designates the current iteration.

Subsequently, CWi counts the number of the running iterations until a postsynaptic spike is fired. Then, the current value of the counter is used to update the synaptic weight value.

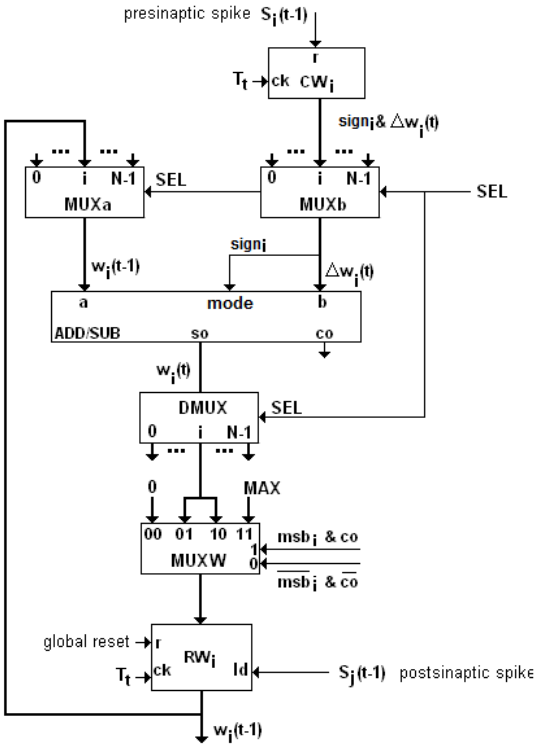

Figure 5. The synaptic circuit.

TABLE 1. THE CODING SCHEME OF THE CWi VALUES.

msb_{CWi} = Δt sign	**$[msb-1....lsb]_{CWi}$ = Δwi**	**change intensity in the synaptic weight**
1 synaptic facilitation	111	✦✦✦✦✦✦✦
	...	…
	001	✦
	000	-- (out of the learning window)
0 synaptic depression	111	✦✦✦✦✦✦✦
	...	…
	001	✦
	000	-- (out of the learning window)

The STDP learning mechanism is implemented by using a peculiar interpretation of the counter value, represented as in the Table I. Thus, the msb bit of the CWi value represents the sign of the Δt time interval defined in Fig. 1. This bit is taken aside and used to control the operation mode of the adder/subtractor circuit. The [msb-1….lsb] bits field of the counter value signifies the change value Δw_i of the synaptic weight w_i. According to this representation, the equation (5) can be rewritten as:

$$\begin{aligned} w_i(t) &= w_i(t-1) + [msb-1...lsb]_{CWi} \quad msb_{CWi}=1 \\ w_i(t) &= w_i(t-1) - [msb-1...lsb]_{CWi} \quad msb_{CWi}=0 \end{aligned} \tag{11}$$

where $[msb-1...lsb]_{CWi} = \Delta w_i$.

With this approach, each iteration, elapsed between the moment of presynaptic spike detection and postsynaptic spike firing, decreases the Δw_i value. The learning window is valid as long as the counting procedure runs. Once the CWi counter hits the zero value, it is considered the learning window is passed. The counter is designed to lock itself in the zero value. Consequently, once the learning window is passed, the counter value remains to zero and the synaptic efficacy is preserved until a new presynaptic spike triggers the counting procedure.

We have chosen a 5 bits resolution for the synaptic weight value and a 3 bits resolution for the change of the synaptic weight. The synaptic weight updating procedure requires a saturation mechanism, that limits the synaptic weight values in [0, WMAX] domain value. The saturation mechanism is implemented by means of the MUXW multiplexer. It controls the source for RWi storing register data input, providing it the maximum value WMAX if an overflow is detected at addition, the minimum value if a borrow is detected at subtraction and the current value of the synaptic weight whenever any of these condition are not met, respectively.

All arithmetic circuit operands are brought to the same length at the layout level, by the concatenation of the '0' bit to the field value in the msb position.

C. The Control Unit

The control unit has two components, a modulo N+2 master counter and a Moore finite state machine. The control unit have to assure the synchronization of the datapath operations, so that the postsynaptic spike in the t iteration to be fired before the arrival of the t+1 iteration, then kept asserted the entire next iteration until soon before the arrival of the t+2 iteration, as is suggested in Fig. 6.

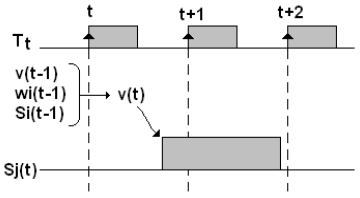

Figure 6. The postsynaptic spike time course.

The Moore finite state machine simulates the neuron states and sends the control signals to the datapath according to the state flow diagram depicted in the Fig 7 and Table II. The finite state machine moves through its states each T_C time period, depending on its input signals.

When the global reset is asserted, the neuron is initialized in the resting state (RST state). After that, the neuron moves into the depolarization, each iteration on. The depolarization consists of two states, called DEP and DEC respectively. If some presynaptic activity is recorded, the finite state machine moves into the DEP state, during the membrane potential is increased. If the presynaptic activity is broken, the finite state machine moves into the DEC state, during the membrane potential is decreased. The firing condition is detected in the SJD state, when the membrane potential reaches the threshold value. Consequently, as soon as the next iteration arrives, the neuron triggers a postsynaptic spike, during the SJ state. After that, the neuron enters into the refractoriness. Initially, the neuron drifts into the absolute refractory period (ARP state), next it go forth to the relative refractory period described by INC and RRP states respectively, during the membrane potential recovers

towards the resting potential. Once the resting potential is attained from below the resting value, the neuron returns to the resting state.

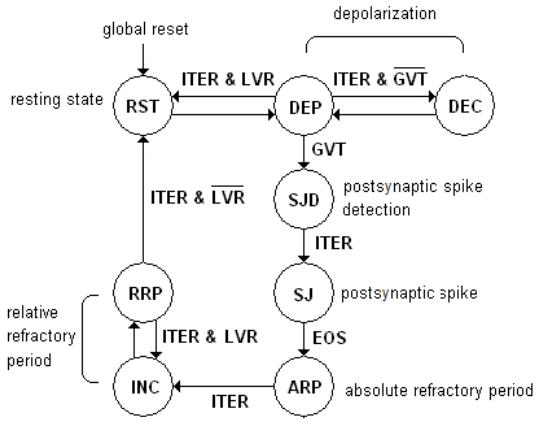

Figure 7. The state flow diagram of the Moore finite state machine.

TABLE 2. THE FINITE STATE MACHINE OUTPUTS.

	after_SJ	set_VR	count	neuron_state	set_SJ
RST	0	1	-	-	0
DEP	0	0	0	-	0
DEC	0	0	1	1	0
SJD	1	0	-	-	1
SJ	1	0	-	-	1
ARP	1	0	-	-	0
INC	0	0	1	0	0
RRP	0	0	0	-	0

The master counter controls the time interval required to compute the membrane potential according to (1). It controls the moment when "set_SJ" control signal is asserted as that the moment of postsynaptic spike firing happens before the next iteration arrival. For this end, the master counter asserts a control unit internal signal, denoted EOS, during the N+1 T_C computation time period, to end the current fired postsynaptic spike before the next iteration arrival.

Also, once a new iteration is running, the master counter generate the SEL signal that sweep all the synaptic inputs, to record the presynaptic activity of the neuron and provide the synaptic weights to the soma circuit.

IV. The FPGA Implementation And Performance Estimation

Two variants of neuron models were implemented into a FPGA device, to verify the logic resource utilization for the hardware neuron implementation and to get the timing information to estimate the time period required to compute the current value $v_M(t)$ of the membrane potential.

The first neuron has a relative small number of synaptic inputs – 8 inputs, and the second one has an appreciable number of synaptic inputs – 30 inputs.

Both models were described in VHDL language [17] and implemented in of Cylone II Altera FPGA devices [18], using the Quartus II design software [19]. During implementation phase, we have used timing constraints to reduce the timing delays of the system signals. The implementation results for both neurons are summarized in the Table III. The usage of the FPGA's logic resources is reported in Logic Elements (LEs). Each LE consists of a combinational function generator, called LUT (Look Up Table) block and a flip flop (FF) register.

TABLE 3. THE LOGIC RESOURCES REQUIREMENTS.

Neuron model	LEs (1LUT + 1FF)	T_{Cmin}
8 synaptic inputs	245	22[ns]
30 synaptic inputs	647	24[ns]

The small input neuron requires 245 Logic Elements (LEs) and the large input neuron requires 647 LEs of 18752 total LEs of the FPGA device, which represents about 1,3% and about 3,45% of the total logic resource provided by the FPGA, respectively. The difference between the quantities of the logic resources is determined by the synaptic circuits, which require 171 LEs for the first neuron and 572 LEs for the second one respectively.

With the recent advance of the VLSI technologies, more logic resource quantity are integrated by the manufactures in the FPGA devices. The growing complexity of FPGA devices allows the implementation of large SNNs. In this respect, a state of the art Stratix V FPGA device may accommodates until 1K neurons with 30 synaptic terminals.

As for the timing, both neurons record similar timings for the minimum T_C value. However, the different number of synaptic inputs makes the current membrane potential value to be computed earliest in 220ns and 768ns, for the first neuron and for the second one, respectively. Consequently, in the proposed neuron model, the iteration can be constrained below to 1μs, even for a relative large number of synaptic inputs.

However, the speed was not one of our primary design goals because of we consider that the computational power of the SNN resides in its biologically inspired information processing. The key contributions of this work are i) the optimized architecture for FPGA implementation towards a reduced consumption of the FPGA's logic resources and ii) the integration of the STDP learning mechanism into the neuronal architecture that allows the learning procedure to be implemented into the chip.

In the last years, various solutions were reported for FPGA based implementation of the digital SNNs. It is difficult to compare these designs due to differences in design objectives and the implementation technologies. Anyway, it is useful to summarize the main approaches reported for the FPGA based SNNs and put our solution into this framework.

One convenient software based approach was reported in [20], were an design environment has been developed in C# language allowing designing SNNs structures with a user friendly interface and converting them into synthesizable VHDL code by means of the VHDL Code Generator tool. However, the proposed the neuron model does not capture the rich dynamical repertoire of the biological neuron. It goes aside the learning mechanism and proposes a simplified model for the soma, without the update mechanism for the synapse, which is described by a simple FIFO based storing element. Moreover, the VHDL code generated by the used tool is far to be optimized for logical synthesis. According to the reported results, a single neuron with a single synapse uses about 202 slices of Spartan 3 Xilinx FPGA device, where the slice represents the Xilinx counterpart for Altera LE. A single slice has 2 LUTs and 2 FFs [21].

A step forward for optimized solution for FPGA based SNNs is presented in [22]. The authors developed a synthesizable VHDL based library for neuronal primitives, used to build up biologically realistic neurons. This approach allows easy portability and usability, but not incorporates the learning mechanism. The authors designed a 86 neurons with 180 synapses SNN which consumed 14192 LUTs and 13166 FFs. Using our neuron model, we estimate that a 80 neurons with 640 synapses SNN, require about 19600 LE.

An interesting software/hardware SNN computing platform is reported in [23], were different stage of the proposed neuron model are processed in parallel, using pipeline techniques. The proposed model capture the soma dynamics into the hardware using a dedicated architecture, but the learning mechanism is left outside the chip, being implemented by means of the software routines. The model is based on a processing element that encloses 6 neurons and requires about 9162 slices of Virtex 2 Xilinx FPGA devices [24].

Using this reference framework, we can visualize the performance of the proposed neuron model and draw some useful conclusions.

V. Simulation Results

To validate the proposed neuron model, an 8 synaptic inputs neuron version has been developed and simulated in the Active HDL design environment, under various test conditions. During the simulation, it were applied random presynaptic spikes at the inputs and were monitored the change of the synaptic efficacy at each synaptic input, the membrane potential variation and postsynaptic spikes firing respectively. It was adopted a 500ns value for the iteration time period and 40ns for the computation time period respectively and 41 successive iterations were recorded to monitor the behavior of the neuron.

The simulation procedure is divided into two parts. The first one, consisted of the 0÷21 iterations, is intended to verify the sequence of the states experienced by the neuron during its stimulation by the presynaptic activity. The simulation results are presented in the Fig. 8.

Initially, all synaptic inputs are initialized to 16, the middle value of the synaptic weights range and the neuron is in the resting state, its membrane potential being at the resting potential (VM=63).

If presynaptic spikes are applied to the synaptic inputs, the neuron starts to depolarize and the membrane potential begins to rise to the threshold value (iterations 1÷5). As long as the synaptic activity is broken (iterations 6÷9), the neuron tends to return to the resting state, the membrane potential decreasing to the resting potential.

During the 12th iteration, the membrane potential reaches the threshold value (VM=255) and a firing condition is accomplished. Consequently, a postsynaptic spike is triggered by the neuron one iteration later (SJ=1). After that, the neuron plunges into the refractoriness (iterations 14÷20).

Initially, the neuron enters into the absolute refractory period, during the membrane potential is forced to the minimum value (VM=0) and the neuron is completely insensitive to any presynaptic activity. Thenceforward, the neuron gets in the relative refractory period, during the membrane potential returns back to the resting potential irrespective of the synaptic activity presence. As can be observed, during the iterations 15÷18, the membrane potential recovers to the resting potential even the presynaptic activity is broken. In the presence of the presynaptic activity, the membrane potential recovering is hastened. However, the neuron sensitivity is lower against the one displayed during depolarization, and the rising of the membrane potential is sluggish. This behavior can be observed monitoring the change of the membrane potential during the 20th iteration (refractoriness) against to the 3th iteration (depolarization). Even each iteration has recorded two presynaptic spikes, the change in the membrane potential during the 20th iteration is half of the change in the membrane potential during the 3th iteration. This behavior consists with the sensitivity coefficient of the neuron, defined in (7). Finally, the refractory period ends and the neuron returns to the resting state (21th iteration) keep waiting new presynaptic activities.

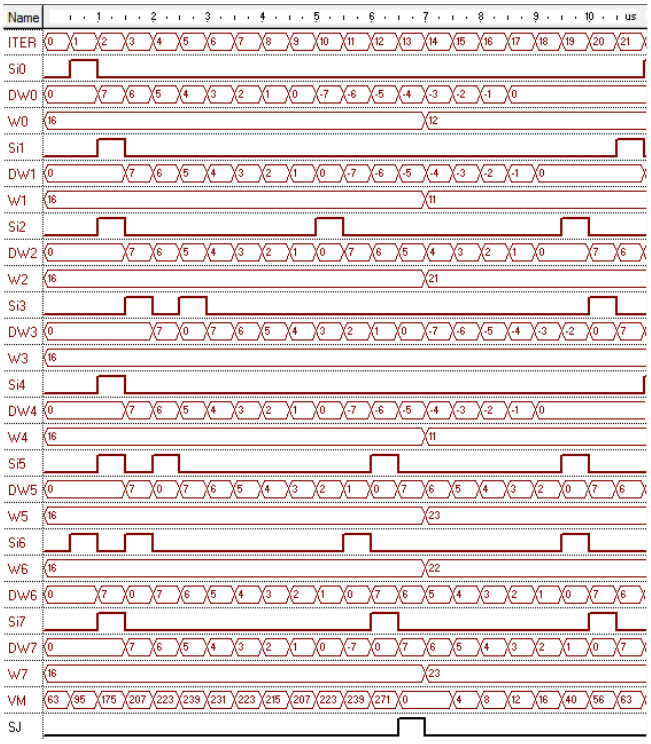

Figure 8. The simulation of the behavior of the neuron.

After the postsynaptic spike is triggered, the synaptic weights are altered according to the SDTP mechanism. Thus, the synaptic efficacy of the inputs with early activity inside the learning window is depressed. This is the case of the synapses 0, 1 and 4 respectively, which have all negative values for the change intensity of the synaptic weight. The synaptic efficacy of the inputs with recent activity inside the learning window is facilitated. This is the case of the synapses 2, 5, 6 and 7 respectively, which have all negative values for the change intensity of the synaptic weight. The synaptic activity of the synapse 3 is outside the learning window. Consequently, the change intensity of the synaptic weight is zero and the synaptic efficacy is unaltered.

The second part of the simulation scenario, consisted of the iterations 21÷40, is intended to point out the ability of the neuron to discriminate presynaptic activity, by means of

the synaptic efficacy modulation mechanism. The simulation results are presented in the Fig. 9.

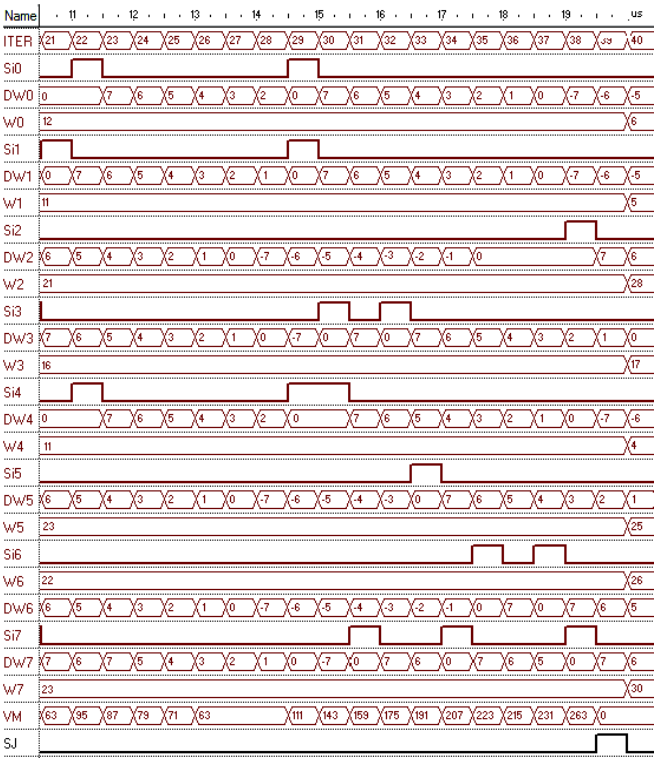

Figure 9. The discrimination of the neuron in the input space.

As can be observed, the information applied to the inputs 0, 1 and 3 respectively, tends to be filtered in time. The neuron has the ability to decide the synaptic activity of these inputs is not correlated with its activity. Consequently, the synaptic efficacy of all these inputs decreases in time toward zero. However, the inputs 2, 5, 6 and 7 respectively, are reinforced. In this case, the neuron decides the presynaptic activity to these inputs has effect on its activity. Consequently, the synaptic efficacy of all these inputs increases in time. The neuron performs a time integration of the information applied at these inputs, firing a postsynaptic spike based solely on this. This behavior can be exploited as coincidence detection inside a time window device.

VI. Conclusion

It was proposed a neuron model suited for digital SNNs implementation, optimized to reduce the logic resource requirements of hardware implementations. The model emulates the basic elements of the biological neuron dynamics, and linearly integrates the contribution of the presynaptic spikes to the current value of the membrane potential, which can fulfill the postsynaptic spike firing condition at most N+2 clock cycles, where N is the number of the synaptic inputs.

The neuron was implemented into a FPGA devices and the obtained results recommend it for 1K SNNs. Due to its performances, the proposed neuron may be used as a computational unit for large reconfigurable SNNs, to explore various network topologies.

References

[1] C. Koch, “Biophysics of Computation: Information Processing in Single Neurons”, New York: Oxford Univ. Press., 1999

[2] W. Maass, “The Third Generation of Neural Network Models”, Technische Universität Graz, 1997.

[3] A.L. Hodgkin, F. Huxley, B. Katz, “Measurements of current–voltage relations in the membrane of the giant axon of Loligo”. *J. Physiol*. vol. 116, pp. 424–448, 1952.

[4] A.L. Hodgkin, F. Huxley, “Currents carried by sodium and potassium ions through the membrane of the giant axon of Loligo”. *J. Physiol*. vol. 116, pp. 449–472, 1952.

[5] A.L. Hodgkin, F. Huxley, “The components of membrane conductance in the giant axon of Loligo”, *J. Physiol*. vol. 116, pp. 473–496, 1952.

[6] A.L. Hodgkin, F. Huxley, The dual effect of membrane potential on sodium conductance in the giant axon of Loligo. *J. Physiol*. vol. 116 pp. 497–506, 1952..

[7] A.L. Hodgkin, F. Huxley, “A quantitative description of membrane current and its application to conduction and excitation in nerve”, *J. Physiol*. vol. 116, pp. 507–544, 1952..

[8] E.M. Izhikevich, “Simple Model of Spiking Neurons”, *IEEE Transactions of Neural Networks*, vol. 14, no. 6, pp. 1569-1572, 2003.

[9] L.F. Abbott, “Lapique's introduction of the integrate-and-fire model neuron (1907)”, *Brain Research Bulletin*, vol. 50, no. 5/6, pp. 303–304, 1999.

[10] K.M Hynna, K. Boahen, “Thermodynamically Equivalent Silicon Models of Voltage-Dependent Ion Channels”, *Neural Computation*, vol. 19, no. 2, pp. 327-350, 2007

[11] J. H. B. Wijekoon, P. Dudek, “Compact Silicon Neuron with Spiking and Bursting Behaviour”, *Neural Networks*, vol. 21, pp. 524-534, 2008.

[12] G. Indiveri, E. Chicca, R. Douglas, “A VLSI array of low-power spiking neurons and bistable synapses with spike–timing dependent plasticity”, *IEEE Transactions on Neural Networks*, vol. 17, no. 1 pp. 211-221, 2006.

[13] M. J. Pearson, A. G. Pipe, B. Mitchinson, K. Gurney, C. Melhuish, I. Gilhespy, M. Nibouche, “Implementing Spiking Neural Networks for Real Time Signal Processing and Control Applications: A Model Validated FPGA Approach”, *IEEE Transactions on Neural Networks*, vol. 18, no. 5, pp. 1472-1487, 2007

[14] P. Arena, L. Fortuna, M. Frasca, L. Patane, “Learning Anticipation via Spiking Networks: Application to Navigation Control”, *IEEE Transactions on Neural Networks*, vol. 20, no. 2, pp. 202-216, 2009

[15] H. Markram, J. Lubke, M. Frotscher, B. Sakmann, “Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs”, *Science*, vol. 275, pp. 213-215, 1997.

[16] D. O. Hebb, “The organization of behavior”, New York: Wiley & Sons, 1949

[17] P.P.Chu, “RTL Hardware Design Using VHDL. Coding for Efficiency, Portability and Scalabilty”, Wiley and Sons, 2006.

[18] “Cyclone II Device Handbook”, available at http://www.altera.com, 2008

[19] “Quartus II Handbook”, available at http://www.altera.com, 2008

[20] A. Rosado-Munoz, A.B. Fijalkowski, M. Bataller-Mompean, J. Guerrero-Martinez, “FPGA implementation of Spiking Neural Networks supported by a Software Design Environment”, *Proccedings of the 18th IFAC World Congress*, 2011

[21] http://www.xilinx.com/support/documentation/data_sheets, “Spartan 3E FPGA device family: data sheet”

[22] J.A. Bailey, R. Wilcock, P.R. Wilson, J.E.Chad, “Behavioral simulation and synthesis of biological neuron systems using synthesizable VHDL”, *Neurocomputing*, vol. 74, pp. 2392-2406, 2011

[23] E. Ros, E.M. Ortigosa, R. Agis, R. Carrillo, M. Arnold, “Real-time computing platform for spiking neurons (RT-spike)”, *IEEE Transactions on Neural Networks*, vol. 17, no.4, pp. 1050-1063, 2007.

[24] http://www.xilinx.com/support/documentation/data_sheets/ds031.pdf, “Vitex 2 FPGA device family: complete data sheet”