DESIGN OF THE SPIKING NEURON HAVING LEARNING CAPABILITIES BASED ON FPGA CIRCUITS [1]

Marek Kraft Andrzej Kasiński Filip Ponulak

Institute of Control and Information Engineering, Poznań University of Technology, Piotrowo 3A, 60-965 Poznań, Poland

Abstract: Hardware real-time implementations of Spiking Neuron Networks (SNN) are wanted for multiple applications. Introduction of the supervised learning mechanism for SNNs is a hot topic. A model of a single spiking neuron having that property is proposed. This is based on LIF simplified model. A number of design issues has been solved in order to enable the correct work of such a neuron during learning phase. The proposed extensions and modifications are described and illustrated with corresponding timing diagrams.

Keywords: Neural Networks, Learning Algorithms, VLSI Design, Biocybernetics

1. INTRODUCTION

Biological neurons process data encoded as spike trains. Input spikes increase the membrane potential. Once it reaches the threshold value, the neuron itself generates a spike on its output. The membrane potential is then reset to the value below the resting potential, and the neuron enters the refractory period. Then the membrane potential slowly rises, until it reaches the resting value again, which also terminates the refractory period (Matthews, 1998). Although the above description is very simplified, it addresses all the basic functionalities of a biological neuron. We may therefore treat such a neuron as a discrete event system, that processes data encoded as spike trains. Depending on the level of biological plausibility, there are a few spiking neuron models to choose from for the implementation. Most widely known are the Hodgkin-Huxley (HH) and Leaky Integrate-and-Fire (LIF) models (Gerstner and Kistler, 2002). The HH model, although offering a very detailed description of the neuron's dynamics, is difficult to implement in hardware because of its functional complexity. An example described in (Ros *et al.*, 2003) uses as much as 7331 slices on Virtex-E FPGA for implementation of only a pair of neurons. This makes the FPGA implementation of larger-scale SNN based on such neuron models very costly and complex. On the other hand, the LIF neuron model is less complex, while preserving fundamental phenomenological properties of the biological neuron. This neuron model is computationally relatively simple, easy to implement in hardware and FPGA-resource-efficient. A LIF neuron model can be referred to a simple circuit consisting of resistors, capacitors, switches and a comparator as it is shown in Fig. 1.

Synapse is represented as a RC filter circuit. Input (voltage) spike passes through it and is converted to a current pulse that charges the capacitor – the capacitor's voltage represents the membrane potential. The membrane potential's change is proportional to the weight of the corresponding input. When the potential reaches the threshold value, an output spike is fired and the membrane potential is reset to a negative (hyperpolarization) value. A resistor in parallel with the capacitor

[1] The work was partially supported by the Polish State Committee for Scientific Research, project 1445/T11/2004/27.

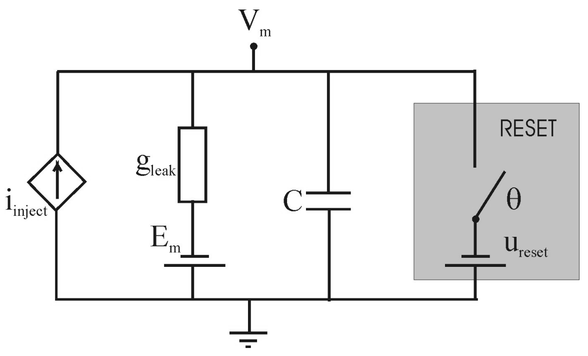

Fig. 1. Simplified circuit of the LIF neuron model.

provides the membrane potential leakage functionality and ensures membrane potential's return to the resting value after hyperpolarization.

Teaching in SNN can be accomplished via synaptic weights tuning. The synaptic weights in the neuron model described in this paper are modified according to ReSuMe learning rule, explained in (Ponulak, 2005; Kasiński and Ponulak, 2005).

ReSuMe is a supervised method for precise learning of spatio-temporal patterns of spikes in SNN. This is still an open issue. According to ReSuMe learning rule, the synaptic weight of an input is increased whenever a spike is transmitted through this input to the neuron shortly before the marked spike time (represented by a triggering spike on the 'tchr' input). The weight of this input is decreased whenever an input spike appears right before the neuron fires (generates the output spike). For the sake of implementation simplicity, the original ReSuMe concept has been slightly modified. The amplitudes of weight changes in our case are determined by the learning windows defined over the difference between the pre- and postsynaptic spike times ($s^l = t^{post} - t^{pre}$) as well as between the presynaptic and the desired spike times ($s^d = t^d - t^{pre}$). For the sake of simplicity the learning windows $W(s^l)$ and $W(s^d)$ modeled in FPGA are implemented as linear functions of s^l and s^d, respectively (instead of exponential functions used originally in ReSuMe learning windows)

$$W(s) = \begin{cases} c \cdot A \cdot \left(\frac{T-s}{T}\right) & \text{for} \quad s \in \langle 0, T \rangle\,, \\ 0 & \text{otherwise,} \end{cases} \qquad (1)$$

where $c = -1$ for $s = s^l$ and $c = +1$ for $s = s^d$, parameter A is a maximal amplitude and T is a width (time-spread) of the learning window. Graphical illustration of the weight update process in ReSuMe is given in Fig. 2.

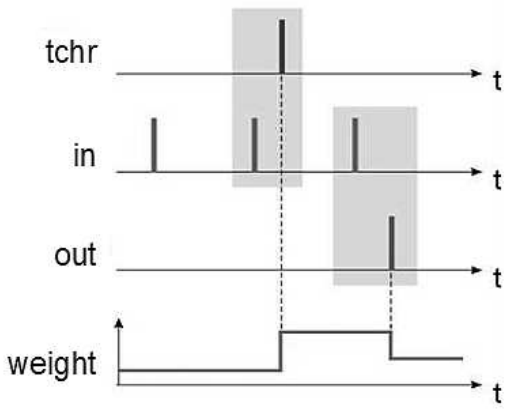

Fig. 2. Graphical presentation of the working principle for ReSuMe learning.

2. SIMPLE IMPLEMENTATION OF LIF

In (Kraft and Kasiński, 2006), our first attempts to model a spiking neuron in FPGA was presented. This implementation lacked the learning functionality, however a 'tchr' input was anticipated to enable the necessary extension in the future. To demonstrate the viability of learning according to ReSuMe in a SNN based on this model was our next goal. The base model was modified for this purpose.

The neuron's inputs are no longer working simultaneously (as in (Kraft and Kasiński, 2006)). In this model the inputs are looked-up sequentially, and the effects of incoming, output and tchr spikes are computed right after. Also, the membrane potential's decay is modeled with a function that approximates the exponential function. The model's work can be divided into some working cycles. Because the inputs are looked-up sequentially, a mechanism is needed to store the incoming spikes for further processing. This is done with two registers - the input latch register and the working register. Each one of them has the width (in bits) equal to the number of inputs. The input latch register is used to latch the incoming spikes on all the inputs during the given working cycle. On the end of the working cycle, this register's content is passed to the working register, and the content of the input latch register is cleared. All the input-related operations are performed on the working register content. Such approach does not cause any loss of incoming data, because the operating frequency of the model is much higher than the frequency at which the spikes are fed to any of the inputs. The content of the working register is looked-up bit by bit, one on each clock cycle. When a logical '1' is met, the corresponding weight value is added to the content of a register representing the membrane potential value. When a '0' is met, no action is taken. Clock cycle count required for the whole operation is equal to the number of inputs. During the next clock cycle,