

FPGA-based Architecture for Real-time Synaptic Plasticity Computation

Bilel Belhadj, Jean Tomas, Olivia Malot, Gilles N’Kaoua, Yannick Bornat and Sylvie Renaud

IMS Laboratory
University Bordeaux 1 – CNRS / ENSEIRB
Talence, France
bilel.belhadj@ims-bordeaux.fr

Abstract—Synaptic plasticity provides the basis for most models of learning, memory and development in neural networks. The challenge for neuromorphic system designers is to find out efficient architectures to process accurately and speedily plasticity rules for a large number of synaptic connections. In this work, we propose a configurable architecture for real-time synaptic plasticity computation. Based on a dedicated plasticity processor, the architecture runs plasticity rules after a predefined configuration. As proof of concept, we implement on a commercial FPGA a biologically inspired form of spike timing-dependent plasticity (STDP) with complex time dependencies between pairs of pre- and post-synaptic spikes. Experimental results evaluate computation accuracy and speed as well as the number of synaptic connections we can process.

I. INTRODUCTION

Synaptic plasticity is one of the important foundations of learning and memory in the brain. There are several mechanisms that cooperate to achieve synaptic plasticity at the synapse level. Much previous work has focused on rate-based Hebbian learning wherein an increase in synaptic efficacy arises from the pre-synaptic cell’s repeated and persistent stimulation of the post-synaptic cell [1]. Recently, the possibility of modifying synapses based on the timing of action potentials has been explored both in neuroscience and neuromorphic engineering disciplines. Spike timing-dependent plasticity (STDP) is a general term referring to functional changes in neural networks and at synapses that are sensitive to the timing of spikes in connected neurons [2].

Spike timing-dependent plasticity rules, in the form of long-term potentiation (LTP) and depression (LTD), provide the basis for most models of learning and memory. This form may be augmented by global biologically inspired processes that carry more behavioral accuracy of neuronal and network activity. In general, such processes involve additional mathematical functions and a number of configuration parameters for each synaptic connection. In order to compute plasticity rules for a large number of synaptic connections, one needs to optimize the calculation of the model and maps it onto powerful and dedicated architecture. This latter should provide the environment of configuration, computation and communication for the entire network.

One digital accurate STDP implementation for small networks (6 analog neurons) has been proposed before [3]. The authors mix software and hardware architectures to configure and process network plasticity. The main consideration then was not large synaptic connections number or speed but flexibility to study and assess STDP features. Other digital architecture extends the functionality of AER communication protocol to implement arbitrary, configurable synaptic plasticity in the address domain [4]. Various forms of learning algorithms can be mapped onto the same architecture by reconfiguring a microcontroller unit. A simple STDP model has been tested for 25 neurons. Experimental results reflect the correctness and robustness of the concept but not the accuracy and efficiency of plasticity processing. In this work, we suggest using commercial FPGA to deal with accuracy/speed compromises in the goal to build an efficient architecture for real-time synaptic plasticity computation.

Our target system can be seen as a series of custom electronic boards which communicate together via a digital bus embedded on a backplane platform, as depicted in Figure 1. All boards are similar and called GAILLIMH daughter boards. Each board employs local architecture composed by 25 analog neurons, a commercial *Xilinx Spartan3™* FPGA, an external memory storage device and a network connector (NC). Neurons are computed on analog ASICs following the Hodgkin-Huxley formalism [5]. FPGAs share out the network plasticity computation. Analog neurons generate spikes at the biological time-scale which imposes a real-time constraint on the plasticity computation delay.

Sections II and III show how the system is configured and how the plasticity is processed. The two last sections evaluate the architecture by implementing a complex STDP model.

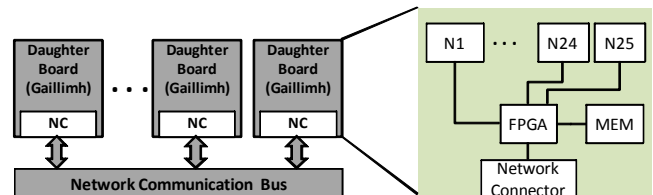


Figure 1. Global system overview: twenty of daughter boards are connected together using backplane communication support

Research was funded by the E.U. Grant FACETS (FP6-IST-FETPI-2004-15879) and E.U. Grant Neurovers-IT (MRTN-CT-2005-019247)

II. NETWORK CONFIGURATION

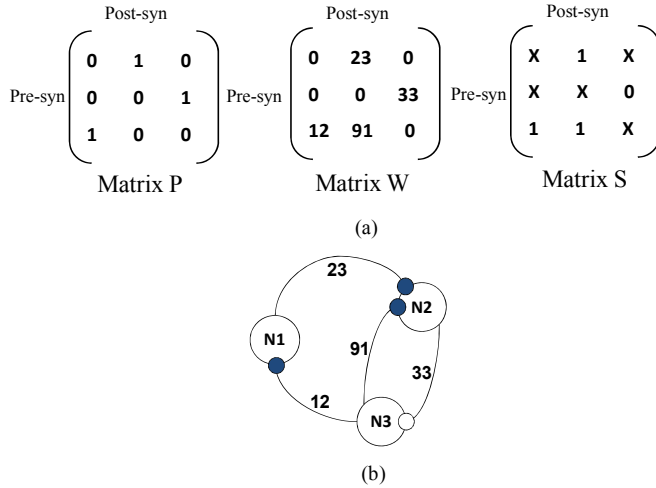


Figure 2. Three matrices for network configuration : (a) example of the configuration of neural network : 3 neurons (b) graphical representation of the neural network. The connections are labeled with their weight values.

Synaptic strength variability is responsible for the plasticity in the network. Only synapses convey information between neurons with or without plastic changes. Plasticity configuration tackles the existence of the synaptic connection, the polarity of the synapse (assuming the connection exists) and the initialization of synaptic strengths. To configure the plasticity of our system, we define three square matrices of N order, where N represents the number of neurons of the network. Each row j refers to the set of post-synaptic neurons of the neuron j . Each column i refers the set of pre-synaptic neurons of the neuron i . Thus, we refer to a connection in the matrix with the couple (row j , column i). The first matrix, $[P]$, indicates whether the individual synapse, binding the pre-synaptic neuron N_j and the post-synaptic neuron N_i , is plastic ($P(j,i)=1$) or not ($P(j,i)=0$). The second matrix, $[W]$, initializes the values of different weights. In the special case when $P(j,i)=0$ and $W(j,i)=0$, the synaptic connection does not exist. Finally, the third matrix, $[S]$, determines synapse polarity: excitatory ($S(j,i)=1$) or inhibitory ($S(j,i)=0$) synapse.

Figure 2 (a) illustrates a configuration example of a three neurons network. Each matrix contains 3×3 elements that correspond to all the possible synaptic connections. In this example, four connections are configured: two are plastic and excitatory $\{(N1, N2), (N3, N1)\}$, one is not plastic but excitatory $\{(N3, N2)\}$ and one is plastic and inhibitory $\{(N2, N3)\}$. Graphical representation of the resulting neural network configuration is given in Figure 2 (b).

III. PLASTICITY ARCHITECTURE

The network configuration maps the topology and plasticity of the network onto neuromorphic target platform. This latter interprets the three matrices described above to process plasticity rules and convey synaptic strengths to the analog neurons. Plasticity architecture is similarly distributed on several GAILLIMH daughter boards where the common digital bus assures communication between them. This section sums up the concept for one daughter board.

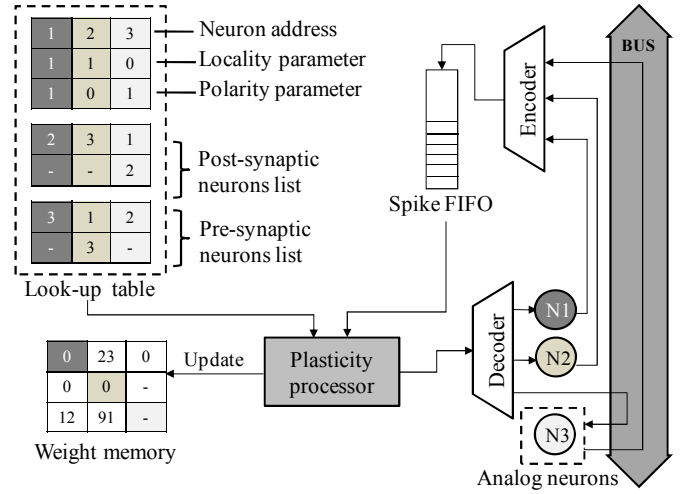


Figure 3. The example of Figure 2 is mapped to the framework by means of look-up table. Neuron N3 is located on another GAILLIMH daughter board.

The architecture employs a look-up table (LUT), dedicated plasticity processor, weight RAM, analog neurons and a number of other additional support circuitry. Figure 3 shows the example of a three neurons network mapped onto the plasticity architecture. Each column in the LUT corresponds to a single neuron; it contains information about neuron address, neuron polarity (excitatory or inhibitory) and pre / post-synaptic neurons lists. Another parameter, called neuron locality, indicates whether the analog neuron is attached to the considered daughter board. We use it to determine if the corresponding post-synaptic connections should be computed locally. When a spike is sent to the system, the sender address is used as an index into the LUT and a signal activates the processor. The plasticity processor scrolls through all the table entries corresponding to the pre- and post-synaptic neurons lists of the sending neuron. Pre and post-synaptic neurons lists enumerate, respectively, pre-synaptic and post-synaptic connections of the sender neuron. Two scenarios are possible; 1) In the case of local post-synaptic neuron, the processor starts by computing plastic rules for this connection, and finishes by updating and sending the new computed synaptic strength to the local post-synaptic neuron. Updated synaptic weight is stored in the RAM weight memory. 2) In the other case (external post-synaptic neuron), the processor just redirects the sender address to the target receiver neuron situated on another GAILLIMH board using the common digital bus. The plasticity rules of this connection will be computed on the receiver daughter board.

Events received by neurons are temporally and spatially integrated by analog circuitry. Each analog neuron receives excitatory and inhibitory inputs that contribute (positively or negatively) to neural voltage variations. Above a certain threshold, a spike is generated and routed to the target board where it will be saved on a specific FIFO queue.

Next sections evaluate the concept of this architecture through an implementation of a complex STDP algorithm. The communication aspects are not considered in this work, only processing features are taken into account. Hence, a dedicated processor is implemented on a *Xilinx*TM FPGA.

IV. STDP IMPLEMENTATION

A. STDP model

The STDP model adopted in this work accounts complex time dependencies between isolated pairs of pre- and post-synaptic spikes [6]. Post-synaptic events that succeed pre-synaptic action potentials in a short duration of time potentiate the synaptic strength (LTP regime), while pre-synaptic events succeeding post-synaptic events by a short duration depress the synaptic strength (LTD regime). The amount of strengthening or weakening is dependent on the exact time of the event within LTP or LTD regime, as depicted in figure 4. The weight update has the form:

$$\frac{dw_{ij}}{dt} = \varepsilon_i \varepsilon_j \left\{ (w_{LTP} - w_{ij}) \sum_{t_i} P[(t - t_j^{last}(t))] \delta(t - t_i) - (w_{ij} - w_{LTD}) \sum_{t_j} Q[(t - t_i^{last}(t))] \delta(t - t_j) \right\} \quad (1)$$

With $P(t) = A_+ \exp(-t/\tau_p)$, $\varepsilon_i = 1 - \exp(-(t_i^{last} - t_i^{last-1})/\tau_{post})$

$Q(t) = A_- \exp(-t/\tau_q)$, $\varepsilon_j = 1 - \exp(-(t_j^{last} - t_j^{last-1})/\tau_{pre})$

where t_j and t_i denote time stamps of pre-synaptic and post-synaptic events. τ_q , τ_p , τ_{post} and τ_{pre} are time constants of exponential decay functions.

The equation (1) summarizes the main features of the model by formulating the time derivative of synaptic weights W_{ij} . P and Q are two “memory” functions that give the amplitude (A_{\pm}) and the sign of the plasticity change, and encode the timing and amplitude of mutual influence between pre- and post-synaptic spikes. Practically, $P(t)$ and $Q(t)$ describe respectively the long-term potentiation and long-term depression as a function of the timing of spikes. The ε terms, called spike efficacy, indicate that the change of synaptic weight is less important for the last occurrence of pre-synaptic or post-synaptic spike than the previous occurrences. Finally, in order to avoid infinite synaptic weight growth, two terms W_{LTP} and W_{LTD} are added so that W_{ij} will be limited by maximum and minimum saturating bounds. These terms are called “saturating factors”, where $W_{LTD} < W_{ij} < W_{LTP}$.

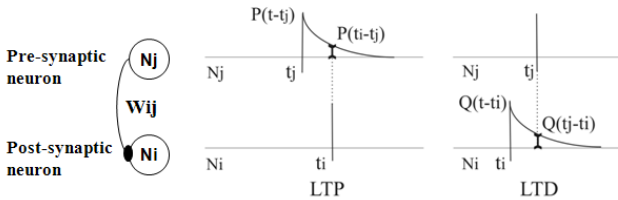


Figure 4. Long-term Potentiation (LTP): post-synaptic spike succeeds pre-synaptic spike. Long-term Depression (LTD): opposite situation.

B. Plasticity processor

The plasticity architecture lends itself to implementations of synaptic plasticity, since information about pre-synaptic and post-synaptic activity is readily available and the content of the synaptic weight fields in RAM are easily modifiable. As in biological systems, synapses can be dynamically created and pruned by inserting or deleting entries in the LUT.

The main purpose of the plasticity processor is to compute STDP rules for each neuron pair of the network. It is designed according to the computational needs of the model described in the previous subsection. Figure 5 illustrates the internal architecture of the processor. Events, stored in the spike FIFO, serve as the trigger that starts the computational cycle. First, the first-out event triggers the PicoBlaze component which schedules and controls computing tasks according to the initial network configuration saved on the LUT. Second, the PicoBlaze distributes and sends computational data for four co-processors. They are implemented to accelerate the computation of the four main functions described in the theoretical STDP model ($P(t)$, $Q(t)$, ε_i and ε_j). They simultaneously process the computation of one plastic connection using the information of the pre-synaptic neuron index and the post-synaptic neuron address provided by the PicoBlaze. The implementation of all co-processors is almost similar; the same block of exponential decay function is instantiated for each co-processor with different time-constant values (τ_q , τ_p , τ_{post} and τ_{pre}). For ε_i and ε_j functions, we add a simple inverter after the exponential block to match with the theoretical equation. The computation of each exponential function leaves hundreds of idle clock cycles when computing two consecutive exponential values. This fact opens perspectives for block exponential reuse during idle cycles which increases the global computational speed of the processor. Therefore, each exponential block is multiplexed in time as many times as the number of idle cycles permits. The RAM memories serve to save intermediate exponential values calculated at different timeslots. Finally, when all the co-processors finish their computation tasks, the PicoBlaze wakes up the weight update block which ties all data received from the co-processors outputs and performs the whole equation (1). When the new synaptic weight is computed and stored in the weight memory component, the same cycle starts all over again.

The next section discusses the speed and accuracy of this implementation and deduces the number of synaptic connections we can process using this processor.

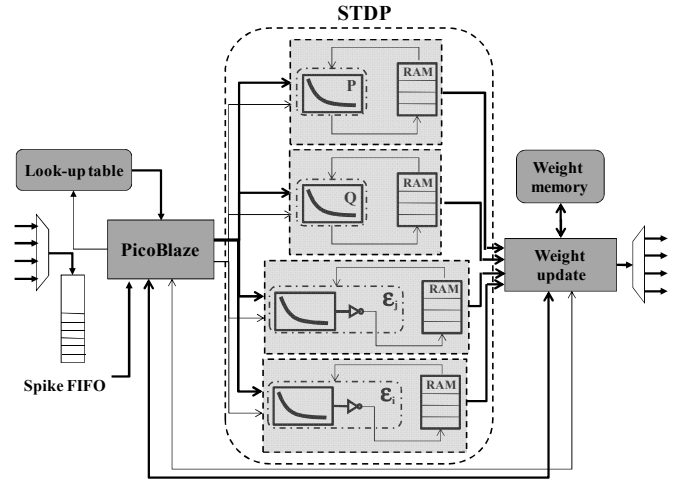


Figure 5. STDP processor architecture. Thick edges represent routed data while thin edges represent computation control.

V. ARCHITECTURE VALIDATION

A. Speed and number of synaptic connections

Speed of the architecture is mostly influenced by the plasticity processor computation delay. To evaluate the performance of the processor, we measured the exact required computation time for different numbers of synaptic connections. This measure is closely related to the values of time constants of the exponential decay functions in the sense that the largest values allow more time multiplexing for exponential blocks. Consequently, the speed of the processor depends on these constants. Figure 6 plots the processor computation delay as a function of the number of synaptic connections for the typical time constant values: $\tau_p=14.8$ ms, $\tau_q=33.8$ ms, $\tau_{pre}=28$ ms and $\tau_{post}=88$ ms [6].

The maximum number of synaptic connections in a network depends on the real-time constraint we define. For example, if we want that the response to a generated neuron event does not exceed 0.8 ms, we can process up to 16,000 synaptic connections (referring to the Figure 6). Furthermore, we can deduce the maximum number of neurons (network size) knowing the density of the connectivity of the network.

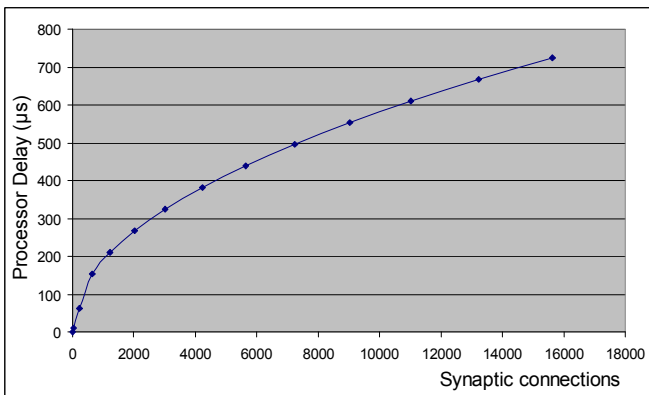


Figure 6. Processor delay evolution as a function of synaptic connections

B. Behavioral accuracy

We have implemented generators of regular patterns for 25 neurons so that each neuron fires approximately with the same frequency. Neurons are phase-locked, with neuron $n+1$ spiking Δt ms after neuron n . In addition, neurons are fully connected between them including self-connections. In total, the network has $25^2 = 625$ synaptic connections. This configuration favors a LTP regime for some synaptic connections and a LTD regime for the others. Since spikes are regularly spread in time and space, we expect gradual evolution between synaptic connections strengths depending on the relative time interval between two spikes of each pair of neurons. These facts are resumed in Figure 7 which plots the evolution of weight strengths as a function of time (s) for all the synaptic connections of the network.

Weight values are computed on 20 bits and coded on the 8 MSB bits ([0:255]). In this experiment, they are initialized to 127 for all synaptic connections. Saturation factors represent

the minimum weight strength ($W_{LTD}=0$) and maximum weight strength ($W_{LTP}=255$) of the weight variation interval. These values directly influence the evolution of the plasticity of the network; the weight strength never exceeds the values of saturation factors. We use the same typical values of exponential time constants given in the last subsection. The rate of the increase of the potentiation is smaller than the rate of the decrease of the depression. This is due to the fact that $\tau_q \approx 2\tau_p$; the value of the depression is greater than the value of the potentiation. Spike efficacy effects are observed in a smooth curved shape of the weight evolutions. However, their accuracy may be enhanced using higher dynamics for computation registers.

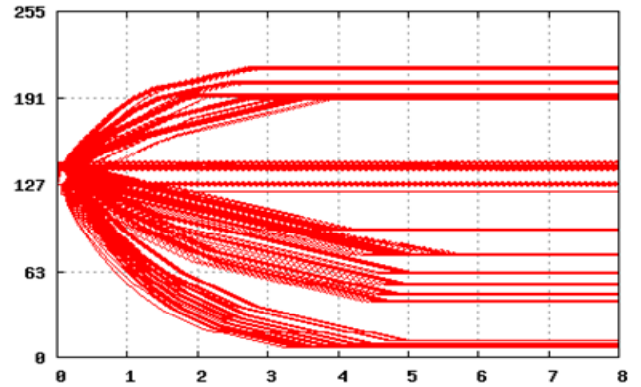


Figure 7. Typical experimental run of synaptic strengths for all-to-all connected 25 neurons: weight strength evolution as a function of time (s).

VI. CONCLUSION

We have designed a dedicated architecture for extensive plasticity processing. The implementation of complex spike timing-dependent plasticity model allows us to validate the architecture efficiency. Unlike classical computing architectures, the proposed architecture performs accurately and speedily plasticity rules computation. However, the architecture may be enhanced by applying other optimization techniques, such as, pipelining the STDP processor.

REFERENCES

- [1] T. J. Sejnowski "Storing covariance with nonlinearly interacting neurons", *Journal of Mathematical Biology*, vol. 4, pp 303-321, 1977.
- [2] P. D. Roberts, C. C. Bell "Spike Timing Dependent Synaptic Plasticity in Biological Systems", *Biol. Cybernetics*, vol 87, pp. 392-403, 2002.
- [3] A. Daouzli, S. Saighi, L. Buhry, Y. Bornat, and S. Renaud "Weights convergence and spikes correlation in an adaptative neural network implemented on VLSI", *Biosignals*, pp. 286-291, 2008.
- [4] R. J. Vogelstein, F. Tenore, R. Philipp, M. Adelerstein, D. H. Goldberg and G. Cauwenberghs "Spike Timing-Dependent Plasticity in the Address Domain", 3rd ed., vol. 2. Oxford: Clarendon, pp.68-73, 2002.
- [5] S. Renaud, J. Tomas, Y. Bornat, A. Daouzli and S. Saighi "Neuromimetic ICs with analog cores: an alternative for simulating spiking neural networks", *ISCAS07, New-Orleans, USA, May 27-30*, pp 3355-3358, 2007.
- [6] A. Destexhe, Z. F. Mainen, "Plasticity in Single Neuron and Circuit Computation", *Nature review Neuroscience*, vol. 6, pp. 789-795, 2004.