Reconfigurable Logic-Based Processor for the Simulation of Neurobiological Processes

Kousanakis Emmanouil

Thesis Committee

Prof. Apostolos Dollas

Prof. Dionisios N. Pnevmatikatos

Dr. Panayiota Poiriazi

School of Electrical and Computer Engineering (ECE)

Technical University of Crete

This dissertation is submitted for the degree of

Master of Science

Chania,Crete

March 2017

I would like to dedicate this thesis to my family

Acknowledgements

I would like to thank my supervisor Professor Apostolo Dolla for his trust, continuous support and guidance throughout the years of this master thesis. In addition, i would like to thank Dr. Panayiota Poirazi and her lab for the excellent cooperation and guidance in the field of neuroscience all these years. Also, i would like to thank Professor Dionysios Pnevmatikatos for his interest in my thesis and his participation in the examination committee.

Also I would like to thank all the MHL members for those two wonderful years that we had together in the lab. Especially, i want to thank Pavlo Malakonaki and Christo Rousopoulo for their useful advices and suggestions regarding my thesis.

Special thanks to Athina Kalampogia for her unending patience and support all these years.

Last but not least, I would like to thank my family for their love and encouragement during all these years.

This work was partially supported by EU H2020 project “COSSIM- Novel, Comprehensible, Ultra-Fast, Security-Aware CPS Simulator”, Project Reference 644042 and by EU FP7 project "QualiMaster", Project Reference 619525.

Kousanakis Emmanouil
March 2017

Abstract

Neuromorphic computing is expanding by leaps and bounds through custom integrated circuits (both digital and analog), and large scale platforms developed by industry and by government-funded large projects (e.g. TrueNorth and BrainScaleS, respectively). Whereas the trend is for massive parallelism and neuromorphic computation in order to solve problems, such as those that may appear in machine learning and deep learning algorithms, there is substantial work on brain-like neuromorphic computing with a high degree of precision and accuracy, in order to model the human brain. In such a form of computing, spiking neural networks (SNN) such as the Hodgkin and Huxley model are mapped to various technologies, including FPGAs. In this work, we present a highly efficient FPGA-based architecture for the detailed hybrid Leaky Integrate and Fire SNN that can simulate generic characteristics of neurons of the cerebral cortex. This architecture supports arbitrary, sparse O(n2) interconnection of neurons without need to re-compile the design, and plasticity rules, yielding on a four-FPGA Convey 2ex hybrid computer a speedup of 823x for a non-trivial data set on 240 neurons vs. the same model in the software simulator BRAIN on a Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz, i.e. the reference state-of-the-art software. Although the reference, official software is single core, the speedup demonstrates that the application scales well among multiple FPGAs, whereas this would not be the case in general-purpose computing approaches due to the arbitrary interconnect requirements. The FPGA-based approach leads to highly detailed models of parts of the human brain up to a few hundred neurons vs. a dozen or fewer neurons on the reference system.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

ANN artificial neural networks

BCM Elie **B**ienenstock, Leon **C**ooper, and Paul **M**unro in 1982

CNS Central Nervous System

BRAM Block RAM

FPGA Field Programmable Gate Array

LTD long-term depression

LTP long-term potentiation

SNN Spiking Neural Network

Chapter 1

Introduction

Research in the field of biology has accumulated a huge amount of detailed knowledge about the structure and the function of the brain. The most important responsibilities of the neuron system are the communication, transfer and processing of relevant information. The central processing unit of the nervous system comprises of neurons, which are connected in a complex pattern. A complex neural network example presents in the Figure 1.1, as the Ramon y Cajal, the father of the modern neuroscience, observed through the microscope in 1899. The drawing shows the different shapes of neurons and the huge extensions of "wires".Neurons and their connections create a dense network of more than 85 billion cells and several kilometers of 'wires' per cubic centimeter. Different regions of the brain are characterized by different network properties. In particular, the key elements, the neurons, differ in shape and type of transmission (excitation or inhibition). Neuron is the basic working and anatomic unit of the brain that transmits electrical signals from one side of the cell to the other. The communication between neurons is achieved through synaptic connections, whose number isn't fixed. Each neuron has approximately 10^4 synaptic connections. The biological neurons and their interconnections compose the neural networks [13].

Researchers were inspired by the way the brain processes information and they developed various artificial neural network models [30]. They attempt to understand the complex process that take place in the dense networks of the central nervous systems in humans and animals and create similar models that can solve problems in the same way that the human or brain would. These models are successfully used in several fields as such as Stock Market Prediction, Medical Diagnosis, Pattern Recognition et al. However, the saturation in the field of artificial neural networks research (ANN) has led to the necessity of deeper understanding the field of biological neural networks and the process of information in biological networks. The main goal of a detailed modeling of neurons, contrary to the simplified ANN models (e.g. Hopfield ANNs [24]), is to apply the knowledge of biological neurons towards the improvement of

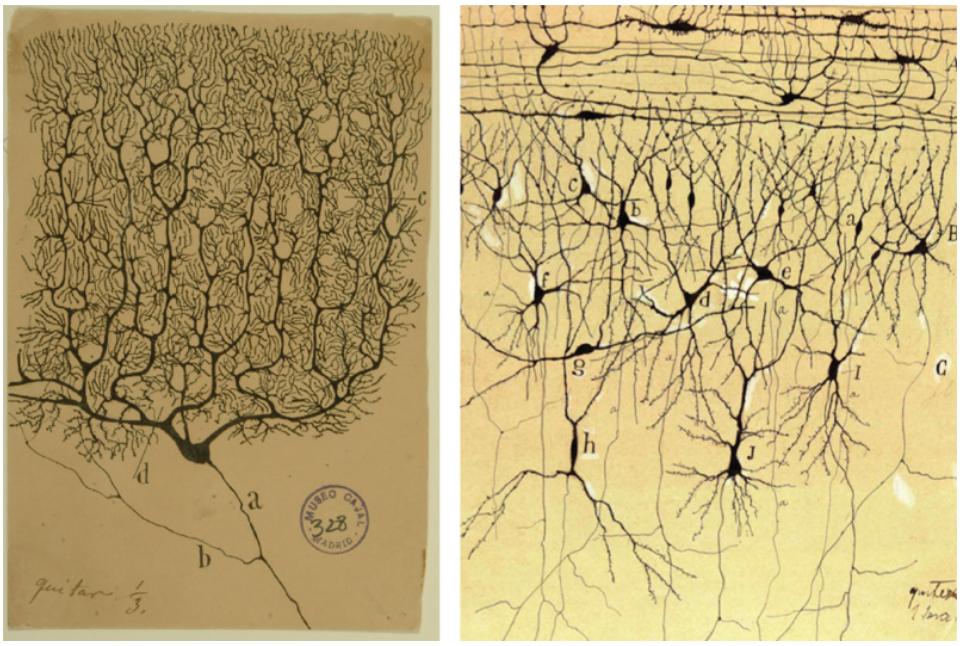

Fig. 1.1 Drawing of the cells, y Cajal [48]

existing techniques and in order to understand in detail how the brain works. The result was the development of several computational models [13], such as the Hodgkin and Huxley [23], the leaky integrate and fire and the Izhikevich model [25], in order to imitate the biological behavior of neurons and the creation of important biologically inspired neural networks.

1.1 Contribution

This thesis presents the implementation of a biological neural model, according to the analysis of the Leaky Integrate and Fire model as a two-layer neural network. We focused on this model because it analyzes in detail the level of synapses and dendrites and can simulate neuronal characteristics that are generic in different parts of the cerebral cortex. The problem is not as much computationally demanding, as it is a memory and communication-intensive problem, due to the detail of neurons and their complex interconnection. In this work, we developed the architecture for a simulator that is an extension-optimization of the work achieved throughout my diploma thesis [27], where simple functions of the SNN model were presented. In particularly, in the diploma thesis, we developed a system that could simulate partially-connected networks with up to 70 neurons, 64 dendrites per neuron and 512 synapses per dendrite. The simulator could simulate excitatory and inhibitory synapses which had different activation curves and the simulation time step was 1ms. The neurons calculate their state("fire" or not) with a simple procedure. At first, they sum their synaptic electrical output at the

dendritic layer and the result is compared with the dendritic threshold. Afterwards, at the somatic layer, they sum their dendritic electrical output and the result is compared with the somatic threshold. This procedure doesn't take into account complicated dendritic, synaptic or somatic mechanisms, such as synaptic activation delay, background noise, refractory period et al. In this work, we extended our model by increasing the level of biological accuracy and implementing additional biological mechanisms. Furthermore, we took advantage of the re-configurable logic and the high memory bandwidth of the Convey HC-2ex and improved the performance level of our model. In the diploma thesis , the system had occupied the total of the available BRAMs, so each further extension or optimization of this work required to re-design the architecture, achieving more detailed neuromorphic model in the same available resources. The contributions of this thesis are:

- In biological level
  - The ability of the system to simulate up to 240 neurons
  - The support of plasticity mechanisms, BCM rule and Homeostatic plasticity that are implemented for the first time in reconfigurable logic
  - The support of background synaptic noise
  - The support of dendritic exponential decay
  - The support of synaptic activation delay
  - The support of neuron refractory period
  - The support of inhibitory and excitatory neurons that have different morphology and connectivity
  - The support of Poisson and Uniform distribution at the neuronal interconnection
  - Simulation time step 0.1ms
- In performance level
  - The extension of the simulator in 4 FPGAs of the hybrid platform
  - The characteristic synaptic curves are modeled with polynomials
  - The system uses all the 16 Memory Controller of the hybrid platform
  - We map deterministically defined but random (i.e. without regular structure or pattern) interconnections into a well-defined memory structure that can be used to stream data from an external memory

  - The simulator is flexible because the external memory feeds the deep pipeline, within the FPGA, with the required data and can be initialized to different interconnection schemes without requiring system synthesis
  - The system was fully developed, implemented, and evaluated experimentally
- The system is fully integrated in the graphical user interface for the parameters input and visualization of the results
- A complete fully configurable neuromorphic simulator

1.2 Publications

- **E. Kousanakis**, A. Dollas, E. Sotiriades, I. Papaefstathiou, D. N Pnevmatikatos, A. Papoutsi, P. C. Petrantonakis, P. Poirazi, S. Chavlis and G. Kastellakis, “An Architecture for the Acceleration of a hybrid Leaky Integrate and Fire SNN on the Convey HC-2ex FPGA-Based Processor”, Proceedings of the 25th IEEE International Symposium on Field-Programmable Custom Computing Machines (FCCM), April 30 - May 2, 2017, Napa, CA, USA
- **E. Kousanakis**, A. Dollas, E. Sotiriades, A. Papoutsi, P. C. Petrantonakis and P. Poirazi, “An Architecture for the Acceleration of the Hodgkin and Huxley Spiking Neural Network Model on the Convey HC-2ex FPGA-Based Processor”, 10th HiPEAC Workshop on Reconfigurable Computing (WRC), January 19th, 2016, Prague
- **E. Kousanakis**, D. Mytakis, A. Dollas, K. Mania, P. Poirazi, E. Sotiriades, A. Papoutsi, P. C. Petrantonakis and I.Papaefstathiou, "Reconfigurable Hardware SNN Acceleration Based on the Hodgkin and Huxley Model and Interactive Visualization of the Results", DENDRITES 2016 EMBO Workshop on Dendritic Anatomy, Molecules and Function, June 18-21, 2016, FORTH, Heraklion, Crete, Poster Presentation.
- **E. Kousanakis**, E. Sotiriades, A. Dollas, P. C. Petrantonakis, A. Papoutsi and P. Poirazi, "Towards an FPGA based Neuroelectronics Emulator", DENDRITES 2014 4th NAMASEN Training Workshop, July 1-4, 2014, FORTH, Heraklion, Crete, Poster Presentation.

1.3 Thesis Outline

The rest of the paper is organized as follows: Chapter 2 provides a background of the biological terms and models, the classic biological neural networks and some of the most known spiking neural networks. In addition, this chapter discusses related FPGA implementations of spiking neural networks. Chapter 3 presents the computational model that we implemented and describes the modeling of biological neurons, their functions and their inter-connectivity. Chapter 4 provides the hardware implementation of the computational model and the integration in the hybrid platform. Chapter 5 shows the experimental setup and the results of this work, while Chapter 6 suggests future improvements of this work and concludes the thesis.

Chapter 2

Background

This chapter analyzes all the biological background for further understanding of this thesis, including the neuron anatomy and function. We, also, present the related work in hardware and software in the field of spiking neural networks. The basic specifications of the hybrid-platform Convey as well as the system platform that our system integrates, are analyzed in this chapter.

2.1 Biological Neuron Structure

The most important responsibilities of a neuron system are the communication, transfer and processing of relevant information. The neuron is the basic functional unit of the nervous system that processes and transmits information through electrical and chemical signals. The neurons receive incoming information from other neurons and send a signal to other neurons, muscles or glands based on their information.The human brain consists of about 10^{11} interconnected neurons that differ from each other. The neurons communicate with each other, through synaptic connections whose number isn't fixed. On average, each neuron has about 10^4 synaptic connections. Most of the neuronal interconnections appear to be created random or based on statistical nature. Nevertheless, the interconnections are created by high precision, both in cell-to-cell level and in entire system level. A number of neurons with their interconnections form a neural network. The whole system of neural networks in the human body is the CNS (Central Nervous System). This system extends throughout the human body with basic parts the brain and the spine. Neurons extend throughout the body, as well. The total number of these neurons and their connections in the nervous system, justify the complexity of the brain as well as the enormous opportunities it presents. Neurons have many different shapes and sizes but a typical biological neuron (Figure 2.1) consists of four major regions: a soma, dendrites, an axon and synapses.

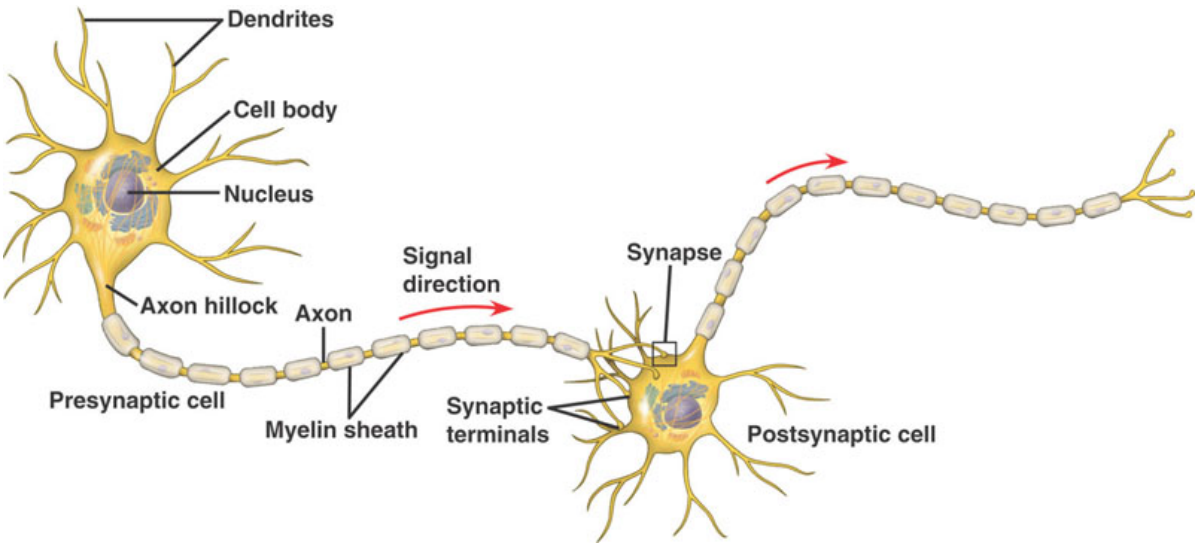

Fig. 2.1 Biological neuron structure [5]

The soma is the body of the neuron and contains all the genetic material of the organism, the cell nucleus. It is the compartment of the cell where that most pronounced effect of the synthesis of enzymes, proteins and other molecules essential for life, occurs.

The axon is a thin fiber that may be up to tens of thousands of times greater in length than the diameter of the body. This structure carries the nerve signals from the neuron. Each neuron has only one axon that can be strongly connected, thus the communication is achieved with many target cells. The connection point has important elements that are needed for the transfer of information to other dendrites of other neurons. Most of the axons are insulated with myelin sheaths.

Dendrites are thin extensions of the cell body and represent the connection point through which the neuron receives input signals from other cells. Each neuron has many dendrites with many branches. Synapse is the point of contact between two neurons. It's a tiny gap at the dendrite that allows the neuron to pass an electrical or chemical signal to another neuron.

2.2 Neuronal functionality

The main purpose of a neuron, in a neural network, is to receive all incoming signals from other neurons, to process and transmit the processed signal to others, spreading the signal to a large network of neurons. A typical neural cell can receive input signals from more than 10,000 other neurons through their axons. The input signals are electrical signals that consist of several Millivolts. Each neuron has two possible states(active and non-active state) which can be located. The voltage pulse at the soma, generated when the neuron is stimulated electrically

by others, is called nerve impulse or action potential or spike and it means that the neuron has been activated. It is a transient, regenerative, electrical thrust, during the production of which the membrane potential of a cell rapidly increases to a value that is approximately 100mV more positive than normal, negative, resting potential. This causes propagation of a signal along the neurons at a great distance. This signal travels between neurons through the neural network without reduction. The maximum spike rate is about 1000 pulses per sec. Spikes cause the propagation of a signal along the neurons and it is the main means of information transfer within neural networks. Therefore, networks that are created by connecting neurons together are called Spiking Neural Networks (SNN).

The electrical signals are created due to the membrane of the neuron that generates a potential difference between the inner and outer surface, just as in a capacitor. The negative potential is usually generated on the inner surface. This is due to the presence of protein molecules with a negative charge which cannot pass the membrane and emerge outside the cell. When the cell is in balance, without the transmitted signal, then the "resting potential" is about -70mV. As shown in Figure 2.2, the membrane potential creates a real short-term evacuation, but gradually returns to its normal state, the potential rest. It is especially noteworthy that during recovery of the neuron is impossible to accept another stimulation, even if multiple signals arrive at the same time and try to stimulate it. This period is called refractory period.

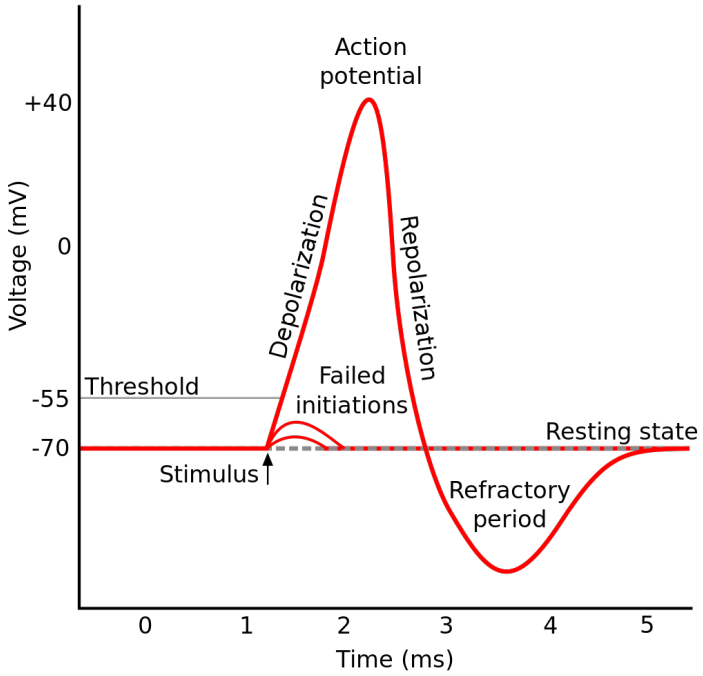

Fig. 2.2 Action potential

At a given time, all the signals that arrive at a neuron are summed. If the sum of signals reaches or exceeds a given value (firing threshold), then it is assumed that the neuron is in an excited state and triggers a fire. In case the sum is less than that given value, nothing happens. The neuron remains inactive and the potential is lost. The threshold value is called firing threshold. The input signal can be excitatory or inhibitory. When the incoming signal is an excitatory that means that the signal is positive and makes the potential of the neuron to approach the threshold. If the incoming signal is an inhibitory, then the opposite thing occurs. The signal is negative and makes the potential to be away from the threshold. The result depends on the function of the threshold. If the signal is equal to or exceeds the threshold voltage, then the neuron is stimulated and sends a pulse that has always the same size. The neuron returns to its original state immediately after the pulse. If the conditions permit, it could be activated again. The signal that was broadcast, continues the same process in other neurons of the network without reduction. The transmission is always in a direction that is away from the cell body.

2.3 Synaptic Plasticity

During our life, our brain is constantly changing. This ability of the brain to change is called plasticity. Of course, brain does not change as a whole, but the individual neurons are modified for different reasons such as our brain development until we reach some age, in response to brain injury or even during learning procedure. There are various mechanisms of plasticity, of which the most important is synaptic plasticity - the science of how neurons alter their ability to communicate.

Synaptic plasticity was first proposed as a mechanism for learning and memory on the basis of theoretical Hebb [20] analysis. The plasticity rule proposed by Hebb postulates that *"When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, in increased."* Theoretical and experimental analysis indicates that not only the Hebbian-like synaptic potentiation is necessary but also the depression between two neurons that are not sufficiently co-active (Sejnowski [40](1977), Stent [43](1973)). Depression is necessary for several reasons. One of them is to prevent all synapses from saturating to their maximal values and thereby losing their selectivity as well as to prevent a positive feedback loop between network activity and synaptic weights. The experimental correlates of these theoretically proposed forms of synaptic plasticity are called long-term potentiation (LTP) and long-term depression (LTD). Two broad classes of models of synaptic plasticity can be described:

1. Phenomenological models: These are very simple models that are typically based on an input-output relationship between neuronal activity and synaptic plasticity. Phenomenological models are typically used in simulations to account for higher level phenomena such as the formation of memory, or the development of neuronal selectivity. The phenomenological models are separated into two different classes, the rate based (Oja [34](1982), Covariance rule, BCM rule [2](1981)) and the spike based, which differ in the type of their input variables.

2. Biophysical models: These more detailed models incorporate more of the cellular and synaptic biophysics of neurons, and are typically used to account for controlled synaptic plasticity experiments [18].

2.4 BCM rule

BCM rule Bienenstock et al. [2] refers to the theory of synaptic modification first proposed by Elie Bienenstock, Leon Cooper, and Paul Munro in 1982 to account for experiments measuring the selectivity of neurons in primary sensory cortex and its dependency on neuronal input. It is a very accurate physical theory of learning in the visual cortex. They suggested a rule for which there is an experimental evidence that requires both presynaptic and postsynaptic activity in order to change a synaptic weight. When a neuron triggers a spike, the synapse that is connected with this neuron modify (or not) its synaptic weight. According to the rule (Figure 2.3), the synapse increases or decreases its strength based on the total dendritic amplitude in which the synapse belongs. The BCM model proposes sliding thresholds (E, TH) for Long-term potentiation (LTP) or Long-term depression (LTD) induction. Equation 2.1 is a mathematical expression of the rule,

$$W_j' = \begin{cases} W_j + rateLTP, & d_i > E \\ W_j - rateLTD, & TH \leq d_i \leq E \\ W_j, & d_i < TH \end{cases} \tag{2.1}$$

where rateLTP and rateLTD are the rates for the Long Term Potentiation and Long Term Depression respectively, d_i is the total amplitude of the i_{th} dendrite of the neuron, TH and E are the thresholds, W_j is the synaptic weight of the j_{th} synapse before the rule and the W_j' is the synaptic weight after the rule.

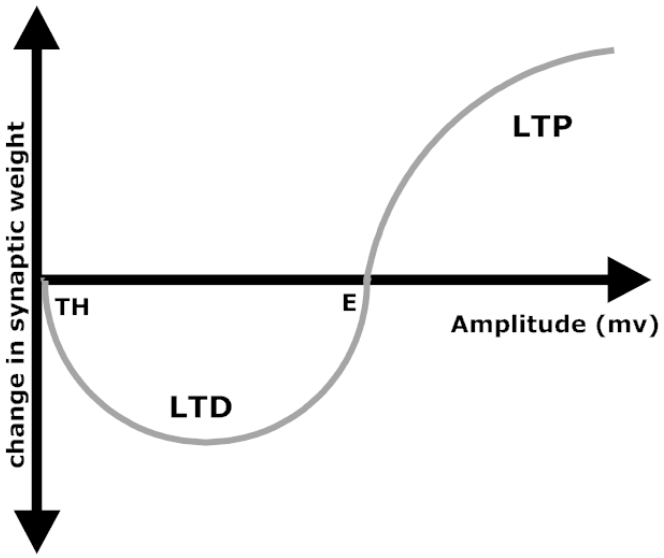

Fig. 2.3 BCM rule

2.5 Homeostatic plasticity

Plasticity rules, such as BCM theory, offer a powerful means of forming and modifying neural circuits. Experimental and theoretical studies have demonstrated that the synaptic plasticity rules do not work alone but in corporation with a mechanism that controls the total synaptic strength of the neurons and stabilize their functions. The mechanism is called Homeostatic synaptic plasticity and is a negative feedback mechanism that the neurons use to adjust their own excitability. The mechanism stabilizes the excessive depression or potentiation of the synaptic strength. Homeostatic plasticity is a slow process that normalizes the synaptic weights. The BCM rule strengthens or weakens the synapses of the neurons, so the homeostatic plasticity works as a mechanism that maintains the stability of neuronal functions. Homeostatic plasticity maintains the total synaptic weight of excitatory and inhibitory synapses based on the equations 2.3 and 2.2:

$$W_j' = W_j \left(1 + \frac{eta * (B_{exc} - A_{exc})}{B_{exc}}\right) \tag{2.2}$$

$$W_j' = W_j \left(1 + \frac{eta * (B_{inh} - A_{inh})}{B_{inh}}\right) \tag{2.3}$$

where $B_{exc/inh}$ is the total excitatory/inhibitory synaptic weight of the neuron before the BCM rule, $A_{exc/inh}$ is the total excitatory/inhibitory synaptic weight of the neuron after the BCM rule, eta = [0,1] is the parameter for the slow recovery of the total synaptic weight.

2.6 Synaptic Noise

Synaptic noise refers to the constant spontaneous synaptic activity of the neurons that produced in the background of a cell. The synapses are dynamically activated without the neuron stimulation of an action potential. The random activation of the synapses are caused because of the quantal release, background activity and/or chemical sensing. Synaptic noise was explicitly simulated by the random activity of inhibitory and excitatory synapses according to Poisson process.

2.7 Our approach

In this thesis, the SSN model has been implemented, based on the analysis of the leaky integrate and fire model, as a two-layer neural network. Our model is a hybrid LIF model because it is implemented with dendrites and can model many mechanism of all the cerebral cortex. Two learning rules (BCM and homeostatic plasticity) have been developed to train the network. In addition, our implementation can simulate synaptic noise, enabling a subset of synapses, based on Poisson distribution.

The hybrid LIF model can simulate excitatory/inhibitory (AMPA, NMDA, $GABA_a$, $GABA_b$) synapses and neurons in detail. Each type of synapse has its own synaptic curve and each type of neuron has its own morphology. More particularly, at the functionality level, a neuron that emits an action potential is often said to "fire". The process of firing takes place at two levels, at the dendritic level and at the somatic level [42]. In order for a neuron to fire, initially, at the dendritic layer, all incoming electrical potentials synapses arriving at a given time are summed. If the total sum has reached or has exceeded a specific voltage value (called the firing threshold), then the dendrite produces a spike and sends the electric potential in the soma. Each dendrite has different firing threshold. This electric potential will contribute to the final depolarization of the neuron. Otherwise, if the sum is less than the threshold, the dendrite doesn't produce an output (it does not "fire") and the potential is lost. The dendrite receives excitatory and inhibitory electrical potentials. If the incoming signal is excitatory synapse, the potential of the neuron is increased towards the firing threshold. Otherwise, if the signal is inhibitory, the potential is reduced from the threshold. The dendritic firing has a 50ms duration and exponentially returns to its normal state. During this time, the excitatory and inhibitory synapses can contribute at the total potential of the dendrite. Therefore, they can increase or decrease the potential up or down of the dendritic threshold.

At the somatic layer, the input voltage of the dendrites that have exceeded the dendritic threshold is summed. If the sum of incoming dendritic spikes is smaller than the somatic

threshold, the neuron remains at rest. Otherwise, the neuron triggers a spike that has a duration of approximately 1.5ms and gradually returns to its normal state, the resting potential. During the refractory period, the neuron cannot produce any spike. In case of the neuron triggers a fire, it activates the connected synapses in a period. The distance of the synapse from the soma is the reason why the activation is not immediate but exists a delay. The values of firing threshold differs at each dendrite and at each neuron. We presented the above information to acquaint the reader to some of the complexities of the model rather than a complete description of it. More detailed information's present in the analysis of the computational model in the chapter 3.

2.8 Related Work

Neural network research has been introduced with the implementation of artificial neural networks several decades ago, for which FPGAs provide a suitable implementation platform. Zhu and Sutton [50] explain the reasons and review the progress that has been made in this research area in recent years. Researchers have developed a variety of artificial neural models in [30] while in [13] great steps have been made towards more realistic biological models. Such advances have led to several detailed biological neural models, which simulate accurately the way in which information is produced and propagated in biological neural networks, such as the brain. On the other hand, neuromorphic algorithms have evolved, which, on the side of biological fidelity are quite abstract. There is a number of biological models [13] such as Integrate-and-Fire, Integrate-and-Fire with Adaptation, Integrate-and-Fire-or-Burst, Resonate-and-Fire, Quadratic Integrate and Fire, Spiking Model by Izhikevich, Fitzhugh-Nagumo, Hindmarsh-Rose, Morris-Lecar, Wilson Polynomial Neurons, Hodgkin-Huxley et al. that aim at a different approach both in the level of detail and in the level of scale.

There are many software simulators such as (GENEGIS [9], SpikeNNS [49], RCS [14], SPIKELAB [17], NEURON [22], BRAIN [15], RSS and SpikeNet [10] that can greatly simulate, realistic biological models and are the state-of-the-art software that biologists use for their experiments. However, large scale simulations have led to hardware implementations in order to achieve simulation times which are better than the real time. In previous years, many computer neuromorphic platforms, based on ASIC chips(Spinnaker [12], Truenorth [31], BrainScales [3], NESPINN [26], MASPINN [39] et al.) and DSP systems (PaSPIKE [47], SPIKE128k [19] et al.) were developed. These platforms can simulate large scale spiking neural networks simulations faster, biological efficiently, with easier accessibility and with less energy consumption. Nevertheless, the ASICs systems aren't flexible because of the fixed architecture that has been made during the fabrication. Thus, there are a lots of FPGA [37] [45] [38] [21] [16] approaches that have been adopted from some of groups because of their flexibility, low cost,

reprogramability et al. In general, hardware implementations made either for high speed simulations for researchers to understand the properties of actual neurons, or to integrate the biological neural networks in computer applications. The ultimate goal is to develop a model equal to the human brain, both in scale level and at the level of detail.

2.8.1 Izhikevich Spiking Neural Network

The Izhikevich Spiking Neural Model [25] not only describes most kinds of "neuron fires" but also models a variety of the neuron's characteristics. The simple implementation and the biological accuracy of the model offers the ability to create large-scale networks. The above factors along with the advantages of FPGA technology have led many research groups to implement this model in hardware [37] [45] [38] [16] [46].

A remarkable work is Moore et al. [32]. They proposed a neuron model using the Izhikevich model. They created a real time SNN in a cluster of 64 FPGAs (Bluehive) which are connected together via SATA connectors, in order to simulate a large-scale model. Each FPGA has 1000 neurons and 1000 synapses per neuron. The neurons' data were stored in an external memory. Initially, the parameters and the synaptic weights are read from the external memory and the equations are calculated. All the neurons that trigger a spike are grouped and their synaptic weights are read from the external memory. The system has a proper synchronization mechanism between FPGAs to update the neurons. They used Bluespec SystemVerilog and the system intergrated in Altera Stratix IV 230 FPGA with two DDR2 memory. They compared their hardware implementation with software they wrote in C language, and they achieved speedup up to 162 times faster than a 4-core Xeon X5560 2.8Ghz server with 48GB RAM. The accuracy of their software was validated based on previous publications. Cheung et al. [7] presented a similar work. The size of the network was the same as before, but the design of this model is focused in the use of the Maxeler platform (using one of the four Virtex6 SX475T FPGA). The spike delivery rate, was the measure of performance and a comparison was made with the GPU NeMo accelerator (Tesla C1060 65nm process), achieving up to 5.27 times faster simulations. In both approaches, the data intensive problem has been solved with event-driven-architecture.

Some researches introduced different approaches of the model such as Thomas and Luk [44] that stored the synaptic data and the neuron spikes in the internal memory of the FPGA. They created a model with 1024 neurons and 1024 synapses per neuron. All the neurons are fully connected to each other, creating different types of spikes (phasic spiking, tonic spiking, and tonic bursting). Their system integrated in a Virtex-5 xc5vlx330, achieving speedup 16 times faster than a 3GHz core2 CPU and 1.1 times faster than 1.2GHz 30-GPU core. A comparison was made with their own software. Cheung et al. [6] based on that work [44]

introduced an event driven architecture. In particular, the state of neurons calculated only when the neuron triggered a spike. Simultaneously, they used systolic architecture that enables different processing elements (PE) to calculate the state of neurons. The implementation integrated in a Virtex-5 XC5VLX155T, achieving 1400 × real time. Also, in comparison with the publication of Thomas and Luk, achieved 148 × real time faster simulation.

2.8.2 Hodgkin and Huxley model

One of the most important models in computational neuroscience is the Hodgkin and Huxley model [23], because it can represent the biophysical mechanisms of the cell in detail. This approach describes with accuracy the characteristics of the real biological neurons but the large number of parameters and the number of equations makes the model a computationally intensive problem.

A noticeable implementation using the Hodgkin and Huxley model is the work that was proposed by Smaragdos et al. [41]. They used an extended HH model for inferior-olive neurons that can simulate a 96-cell network that is fully connected, simulated in real-time speed, and a 1506-cell network simulated in non-real-time. The model is an extension of the original HH based model, and divides the neuron into three computational compartments (soma, dendrite, and axon). The interconnection between neurons achieved through gap junctions. The network is fully connected, so that the number of gap junctions in each neuron is the same as the total number of network's neurons. For a 96-cell network (real-time), a Virtex 7 (XC7VX485T) FPGA achieved speedup of x12.5 against a reference C implementation running on an Intel Xeon 2.66GHz machine with 20GB RAM. For the non-real time implementation of 1056 neurons, they achieved a n FPGA speedup of x45 against the C code.

Furthermore, Beuler et al. [1], has proposed a custom-made simplified model of a HH neuron at real time simulation. The real time achievable network was up to 400 neurons and they included a simple GUI for parameter setup. The spiking neural model focused in a specific area of the brain, so the interconnection did not simulated with dendrites, as the most implementations, but with gap junctions.

Leung et al. [28] developed an FPGA-based spiking neural network to model the early stages of stimulus encoding and processing in the rat whisker system. Their neural circuit simulator (Cyclone II FPGA) can simulate 14 parallel neurons in just 265 ns achieving a 386-fold speedup over the software implementation of the same model.

2.8.3 Integrate-and-Fire

A less complicated approach of spiking neural network is the Integrate-and-Fire. This model is the most popular in this field because it neglects some characteristics and thus can easily model several hundreds of neurons. There are variations of this model (such as Integrate-and-Fire with Adaptation and Integrate-and-Fire-or-Burst) that add additional neuron functions. The best variant of the leaky Integrate and Fire is the Quadratic Integrate and Fire which not only simulates many functions of the neuron but also is ideal for large-scale modeling. Various implementation of integrate and fire model have been developed [8] [11] [35] but they didn't approach the SNN using dendrites and synapses as separate parts of the soma. Some remarkable works are presented below.

On a recent paper of Cheung et al. [8], a simulator is introduced that can simulate various spiking neural network, including integrate and fire, and STDP learning mechanism. They use a high level API to configure the processors with specific parameters and mechanisms, so they succeed speedup and large-scale networks. However, their implementation requires recompilation when the configuration of the model is changed. They developed the simulation platform based on FPGA systems from Maxeler Technology. The system can simulate 600000 neurons in non-real time and 40000 neuron in real-time. Neuronal and synaptic parameters are stored in high-capacity off-chip memory while the others are stored in high-speed on-chip memory to optimize the overall memory latency. Using one FPGA of the Maxeler platform, the system achieved a speedup up to 33.6 times the speed of an 8-coreprocessor, or 2.83 times the speed of GPU-based platforms.

Deng et al. [11] proposed an implementation of the leaky integrate and fire model using feedforward network (FFN) architecture.According to the multilayer FFN structure, the neurons are grouped per eight and each group of neurons communicates only with its neighbor groups to transfer the spikes. The architecture, using a Stratix III EP3SE110F1152C4 device at 50MHz clock frequency, are able to simulate the full-parallel three-layer FFN, so the system can simulate 24 neurons and achieved 50000 times faster simulation time than real time. Pearson et al. [35] present the implementation of an HSNN based on the Oscillatory Dynamic Link Matcher (ODLM) algorithm. The ODLM uses a leaky integrate and fire neuron model to approximate the behavior of relaxation oscillators. A fully connected network of up to 648 LIF neurons and 419904 synapses can be implemented on a single Xilinx XC5V50SXT device, processing up to 6M spikes/s.

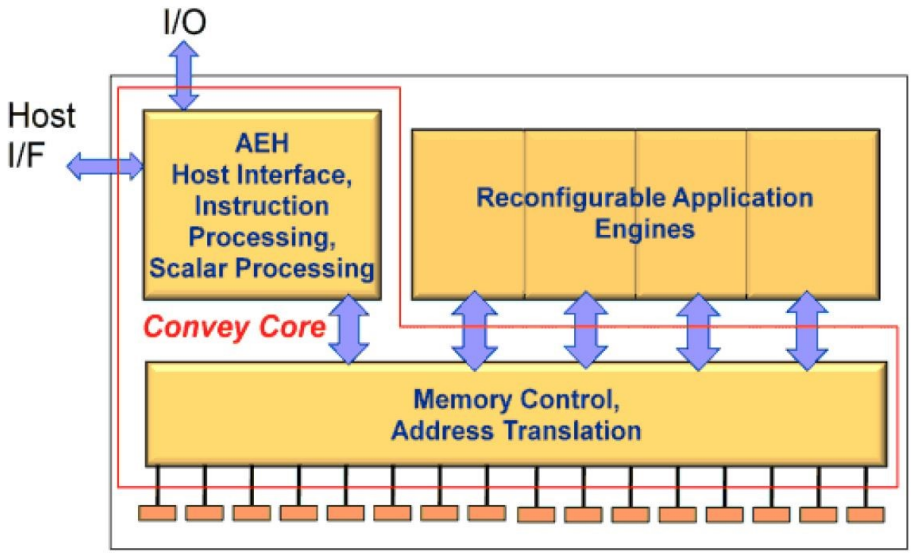

Fig. 2.4 Coprocessor of the Convey platform

2.9 Convey HC-2ex hybrid platform

2.9.1 Introduction

The convey-computers introduced a hybrid platform that is based on reconfigurable logic. The convey HC-2ex is the second generation of the platform and combines an Intel Xeon processor and a Convey designed coprocessor based on Xilinx Field Programmable Gate Arrays, with its own high-bandwidth, virtual memory addressed and cache coherent memory subsystem. It also offers an ANSI standard development environment, increasing productivity and portability.

The coprocessor system Figure 2.4 is based on reconfigurable logic to increase the performance of applications in comparison with the systems based on standard X86. Due to the programmable nature, the hardware allows the architecture to be redefined to fit in the respective applications. These reconfigurable instruction sets are called personalities. The system provides some personalities, such as single-precision, double precision vector personalities, financial analytics personality and Smith-Waterman personality, which can be used to accelerate certain applications. However, some applications require specialized functionality, which cannot be offered by the existing personalities, thus the Convey-Computers designed a framework to enable the development of Custom Application Engine Personalities, including extension of the instruction sets that allow custom execution.

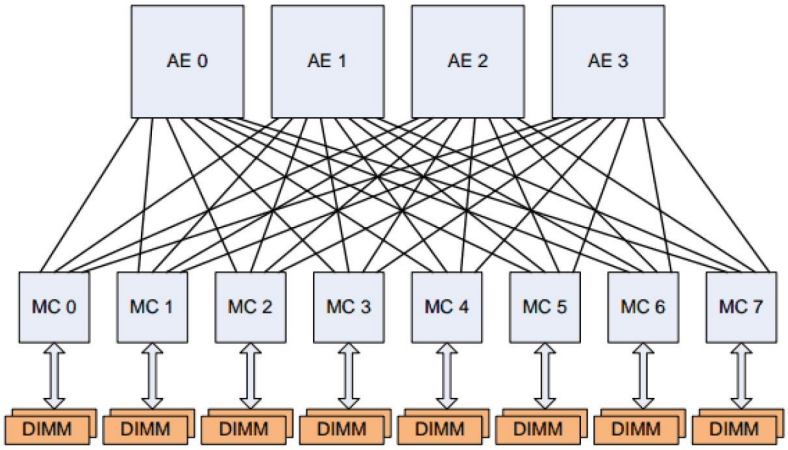

Fig. 2.5 AE-to-MC connectivity on the coprocessor

2.9.2 Coprocessor Architecture

The coprocessor of the platform consists of three main units: The Application Engine Hub (AEH), the Memory Controllers (MCs) and the Application Engines (AEs). Custom commands, which have been developed for the coprocessor, are implemented in Application Engines FPGAs and the AEs are the only FPGAs that are redefined for different personalities. The AEs contain four basic interfaces communicating with the rest of the system: the despatch interface, memory controller interface, the managment/ Debug interface and AE-to-AE interface.

2.9.3 Personalities

The user, in order to design a personality, requires the use of PDK that has:

- A set of Makefiles that support the simulation and the synthesis flow design.
- A set of Verilog files for the communication of processor with the FPGAs
- A set of simulation models for all the non-programmable parts of the coprocessor (memory controllers, memory modules)

2.9.4 Memory Controller interface

The interface Memory Controller (MC) AEs provides a direct access to the coprocessor memory. Each of the 4 AEs is connected to each of the 8 MCs through a DDR interface with a clock frequency of 300MHz. The interface of MC in the AE FPGAs is provided by Convey. Each interface of the MC 8 to AE FPGA is directly connected to a single Memory Controller and all

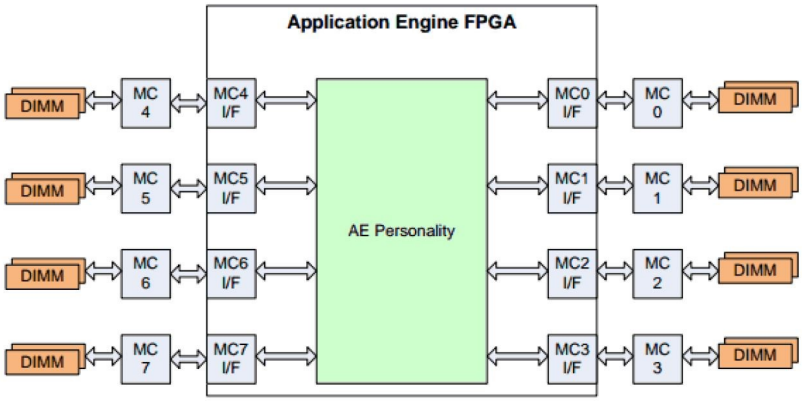

Fig. 2.6 The MC interface connections to the Memory Controllers on the coprocessor

the MC are connected to the 1/8 of the coprocessor memory. The diagram below (Figure 2.5) shows the connectivity between AE-to-MC in the coprocessor.

The Figure 2.6 shows the MC interface connections of the AE with the Memory Controllers on the coprocessor. The interfaces of 8 Memory Controller located on the left and right side of the AE FPGA. Each memory controller is connected with 2 DIMM modules. The AE personality is responsible for decoding the virtual memory address, thus only applications for a particular MC are sent.

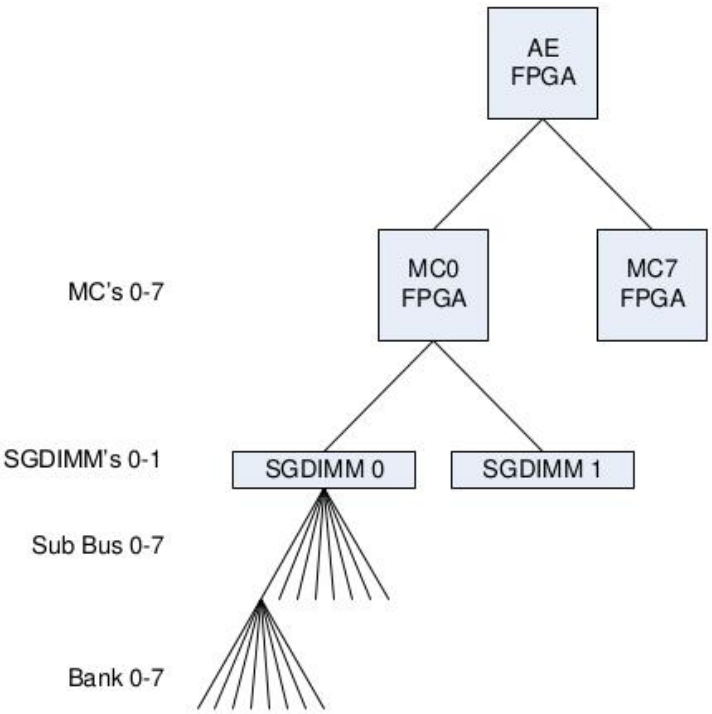

Fig. 2.7 Memory Hierarchy

2.9.5 Memory System

The Convey memory system uses Scatter / Gather DIMMs, which has 1024 memory banks.The banks are spread across 8 Memory Controllers. Each memory controller has two 64-bit buses and each bus has access to eight individual buses (8-bit per sub-bus). Finally, each bus has eight banks. The 1024 banks obtained as follows:

$$MCs * 2DIMMs/MC * 8subbus/DIMM * 8bank/subbus \tag{2.4}$$

The above Figure 2.7 shows the coprocessor memory

Chapter 3

Spiking Neural Network modeling

This chapter presents the modeling of the computational model in order to understand the design of architecture. In addition, this chapter discusses the dimensioning of the problem, noting how it can be customized according to the user preferences, to simulate networks with much fewer neurons, dendrites and synapses or networks with fewer neurons but with hundreds of dendrites and synapses. An abstract description of the whole application(GUI, hardware) is presented, giving the big picture of the simulator.

3.1 Synaptic responses

In this thesis, a spiking neural network, with up to 240 neurons, was implemented. Each neuron consists of up to 64 dendrites and each dendrite has up to 512 synapses. The network is partially connected, i.e. the majority of synapses are connected to a neuron of the network, so a complex network is build that operates in parallel and processes information.

The model on which our design is based, is able to support simulations of different types of excitatory and inhibitory synapses and neurons, implemented in the NEURON simulation environment [22]. In our networks, the distribution of excitatory neurons or synapses is, by default, 80% of the total number of neurons or synapses and the inhibitory neurons/synapses is the remaining 20% [29].Nevertheless, our design is independent from these numbers, because they can be provided as an input to our system. More details are provided in Section 3.4 where the interface of the simulator is presented. Our models can accurately simulate hundreds of synapses, so all synaptic curves were infeasible to be stored on-chip. We approached the problem with two different ways.

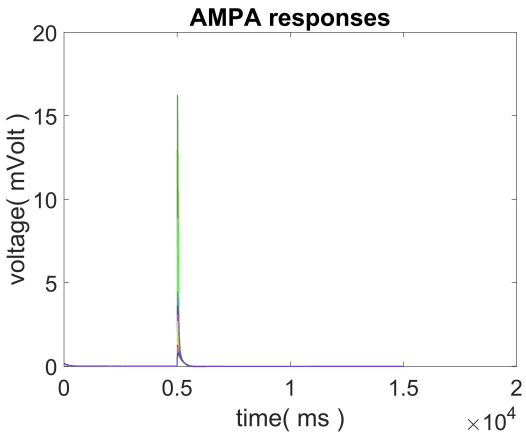

Fig. 3.1 AMPA synaptic responses as they recorded

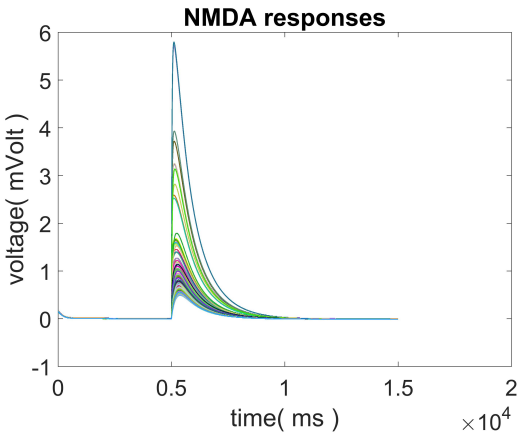

Fig. 3.2 NMDA synaptic responses as they recorded

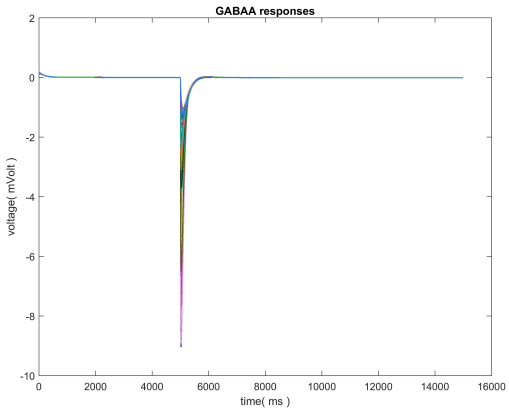

Fig. 3.3 GABAA synaptic responses as they recorded

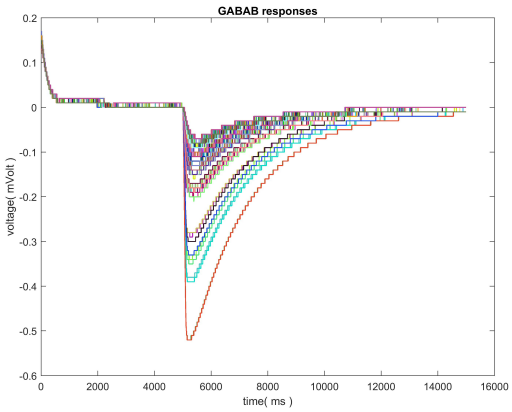

Fig. 3.4 GABAB synaptic responses as they recorded

3.1.1 First approach

In the diploma thesis [27], we simulated synaptic curves (AMPA, NMDA, $GABA_a$, $GABA_b$) to different dendritic branches and recorded the responses. Figures 3.1 3.2 3.3 3.4 show all the activation curves of each type of synapse, as they had been recorded. We observed that individual responses differ in the amplitude and duration. Activation curves of each synapse vary due to different distance and/or morphological properties (e.g. thickness) of dendrites [4]. Thus, we decided to store on chip the shortest and the lowest-amplitude response and re-scale the synaptic weight in each synapse. The combination of these can produce different activation curves for each synapse. Figures 3.5 3.6 3.7 3.8 present the characteristic synaptic curves that are stored in the internal memory of the system.

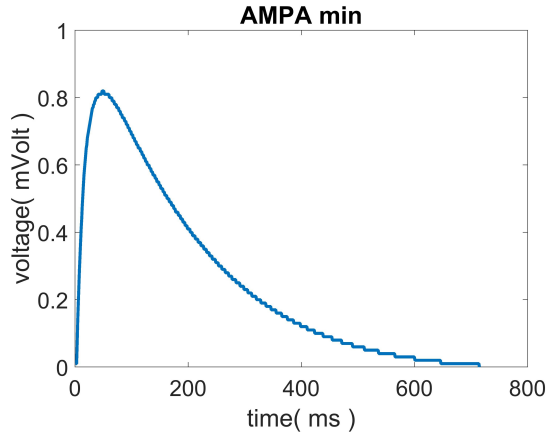

Fig. 3.5 AMPA characteristic synaptic response

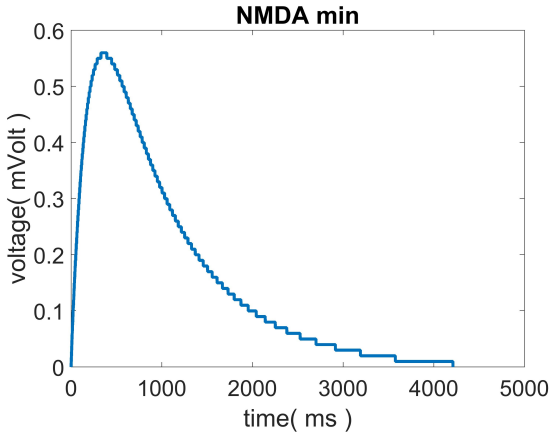

Fig. 3.6 NMDA characteristic synaptic response

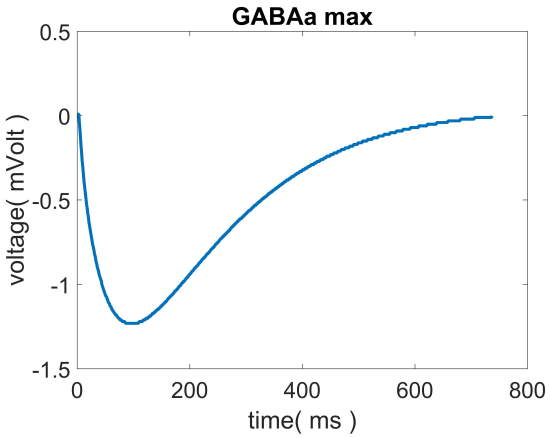

Fig. 3.7 GABAA characteristic synaptic response

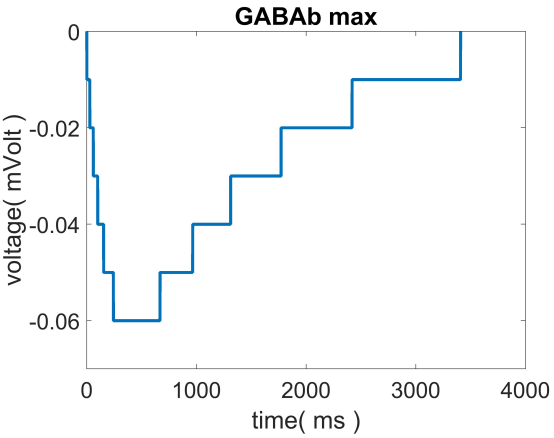

Fig. 3.8 GABAB characteristic synaptic response

Table 3.1 Curve fitting score, as the curveExpert tool shown

Synapse type	AMPA		NMDA		GABAA		GABAB	
	x<73	$x \geq 73$	x<394	$x \geq 394$	x<107	$x \geq 107$	x<668	$x \geq 668$
fitting score	91.4%	97.6%	98.3%	99.1%	97%	99.2%	99.1%	88.4%

3.1.2 Second approach

In this thesis, we approached the solution with a different way. As we will show below, the main limitation of the system is the Block Rams, so we decided to save some of the internal memory. The characteristic responses were studied with the curve fitting technique and were analyzed into a mathematical function. Hence, the calculation of the synaptic output only needs the coefficients of the equations that stored in the internal memory of the system and occupied less resources than the discrete values of the curves.

Curve fitting is the process of creating a curve or a mathematical function that best fits to a set of data. Curve fitting can be done either by interpolation or by smoothing. In the first case, the data is adjusted precisely. In the second case, a smooth function that approximately matches the data is created. One approach of curve fitting process, is the regression analysis which focuses on statistical inference, such as the calculation of uncertainty that presents in a curve that is fitted to data. There are many statistical packages (R), numerical softwares (GNU Scientific Library, MLAB, MATLAB, SciPyand) and specialized programs (table2DCurve, QtiPlot, curveExpert Professional) that offer commands for doing curve fitting. We used curveExpert professional program because it was effective and easy to use.

The technique was to create a polynomial function, which can be implemented relatively easily in hardware. Initially, the result of regression analyses was polynomial of degree higher than 10. However, the available system resources were not enough to solve so high degree equations. The tool extracts better results when we spitted the curves into two parts based on the max amplitude of each response. The results were $2th$ degree polynomials and fitting score 88.4%-99.1% (Table 3.1). The score indicates how well the model adheres to the data. The equations 3.1 3.2 3.3 3.4 show the coefficients of the polynomial for each type of synapse.

$$f_{AMPA}(x) = \begin{cases} 11.9653755x^2 + 4.0361361x - 0.0412964, & x < 73 \\ 120.053571x^2 - 0.3883324x + 0.0003172, & x \geq 73 \end{cases} \tag{3.1}$$

$$f_{NMDA}(x) = \begin{cases} 5.35194429x^2 + 0.4138219x - 0.0006421, & x < 394 \\ 102.816961x^2 - 0.0792575x + 0.0000171, & x \geq 394 \end{cases} \tag{3.2}$$

$$f_{GABAA}(x) = \begin{cases} -19.434211x^2 - 3.4437076x - 0.0210641, & x < 107 \\ -229.06310x^2 + 0.6599898x - 0.0004825, & x \geq 107 \end{cases} \tag{3.3}$$

$$f_{GABAB}(x) = \begin{cases} -1.6955116x^2 - 0.0285431x + 0.0000302, & x < 668 \\ -8.3430815x^2 + 0.0032363x - 0.0000002, & x \geq 668 \end{cases} \tag{3.4}$$

A basic method for calculating the polynomial function is the Horner's rule. The rule is quite simple and simultaneously effective for calculating polynomials. If a polynomial of degree n solved directly without conversion, takes n (n + 1) / 2 multiplications and n additions, making the complexity of O(n2). Based on the rule of Horner, the function can be turned and reduced the number of multiplications to n. The number of addition, as we will see, remains the same. The computational complexity is reduced to O(n), making high degree polynomials more manageable. Therefore, each of the polynomials 3.1 3.2 3.3 3.4 requires 5 multiplications and 2 additions. Considering the polynomial equations stated in 3.1 3.2 3.3 3.4, Horner's rule re-writes them as:

$$F(x) = x(ax + b) + c \tag{3.5}$$

where a,b,c are the coefficients. The Horner's rule uses the smallest number of iterations for the calculation of a polynomial and is easily implemented in hardware because of the power elimination.

3.2 Neurons functionality

Figure 3.9 represents an abstract image of the neural network. The left side shows the plurality of neurons, each of which has an electrical output that occasionally produces a spike. The spike, in our system, is modeled using binary signals. The value of 0 and 1 is assigned to the existence or not of action potential. The synapses are connected to the outputs of other neurons through their axons. If a neuron triggers a spike, the synapses associated with it are activated and

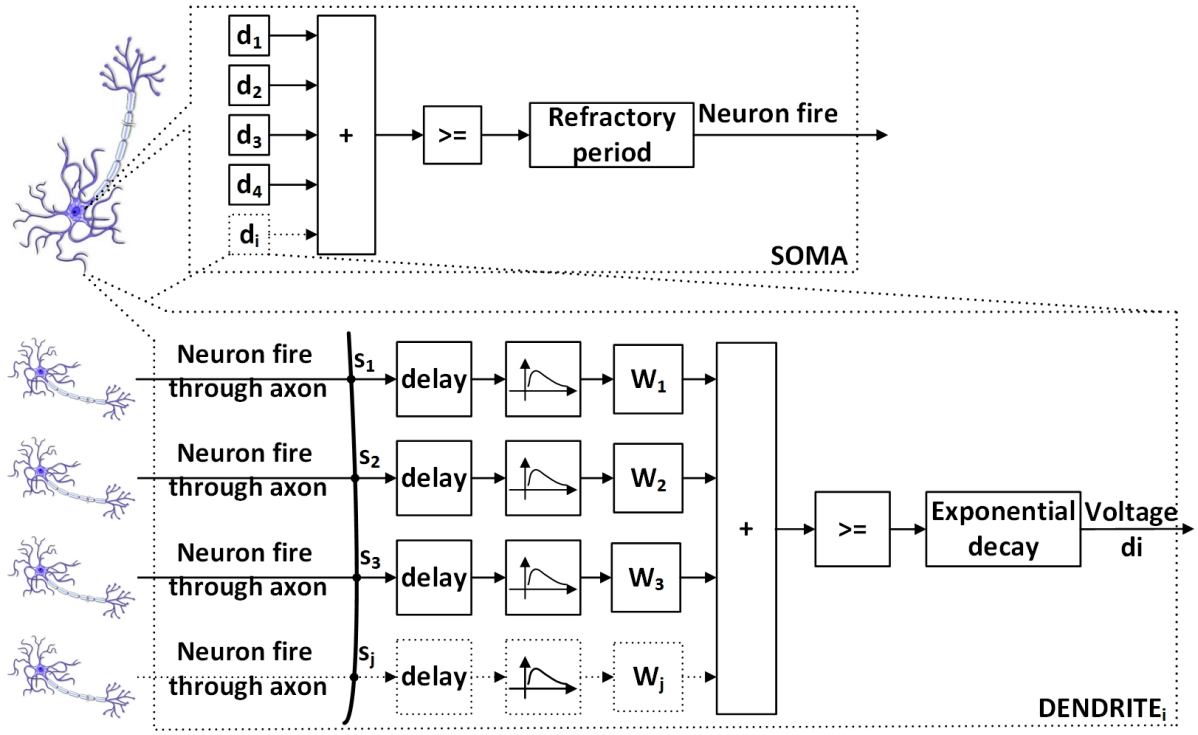

Fig. 3.9 Abstract image of the neural network

produce an electrical output that depends on the synaptic type (AMPA/NMDA/GABAA and GABAB). The synapse is not directly activated because of the distance between pre-synaptic and post-synaptic neuron. We approached this synaptic function with a timer that delays the synaptic activation. When activating a synapse, the model samples the discrete values of the response from the point in which the potential increases from the baseline, and for a time duration that is dependent on the type of synapse. The discrete value of the synapse activation curve is combined with the synaptic weight and the result is the actual electrical output of the synapse. If the neuron that is connected with the synapse triggers a spike again, the sampling starts from the beginning. The output of a non-activated synapse in the baseline, is few mV as shown in figures 3.1 3.2 3.3 3.4. However,due to the fact that the total contribution is negligible, we assume that a non-activated synapse has zero output.

At the dendritic layer, each neuron sums the outputs of its synapses. In case the sum exceeds the firing threshold of the dendrite, it triggers a dendritic spike duration 50ms and the total sum are transferred in the soma of the cell. Each dendrite has a different firing threshold because of its morphology. We are modeled this particularity and defined randomly dendritic thresholds to be in the range of 14mV to 40mV. During the spike period of 50ms, the inhibitory synapses may drop the sum below the firing threshold. In that case, the dendritic amplitude will exponentially return to the baseline. However, incoming excitatory synapses increase the amplitude sum over

the firing threshold to produce a more powerful dendritic voltage. The dendritic exponentially decay mechanism is modeled by multiplying each time step of the dendritic amplitude, with a factor. The dendritic spike duration may vary between different simulations based on the preferences of the scientists, so the factor mentioned above, is calculated at the setup of the system. The equation to calculate the factor of the exponential decay is presented in 3.6.

$$F = 2^{(log(d)/t)} \tag{3.6}$$

where t is the time step, d is the dendritic spike duration that the user sets.

Similarly, at the somatic layer, the dendritic outputs are summed. If the value is greater than the somatic threshold, the neuron triggers a "fire" that lasts 1.5ms. The spike is an enable signal of the interconnected synapses. The excitatory neurons have 30mV somatic threshold and the inhibitory neurons have 20mV somatic threshold, to simulate the higher frequency of inhibitory compared to excitatory neurons [29]. The neurons have a refractory period about 1.5ms. At this time, the soma returns to its resting state, so it cannot receive electrical input from the dendrites. In addition to spikes, neurons also receive external stimuli. Without external stimuli, the system would be in a continuous rest. Certainly, the system can meet different thresholds per neuron and different refractory period time depending on user's preferences.

Our aim was to made a simulator that is similar with the state-of-art simulators and is useful for the scientists. Neuroscientists, group the synapses per 5 or 10 when they perform the simulations. Thus, we implemented the same functionality, grouping the synapses per 5 or 10 randomly. This functionality led design to become more efficient, because in that way we release internal memory and as a result , the simulator can host more neurons.

An additional mechanism of our model is the synaptic noise that randomly activates a set of synapses. Synaptic noise is a background function that exists in the neurons caused by many factors. Our model implements this function and activates some of the inactive synapses at a specific time based on poison distribution. The selection of the synapses is done at the setup of the system when the user sets the frequency of the synaptic noise. The synapses where the synaptic noise mechanism exists, are the unconnected synapses of the network. The synaptic features that discussed above are summarized in eight parameters:

1. neuron address
2. dendritic address
3. synaptic address
4. type of synapse

5. synaptic weight
6. sampling counter
7. activation counter
8. synapse activation

3.3 Interconnectivity

The biggest challenge in the development of the biological model was the interconnectivity of neurons. The problem at hand is not computationally intensive but rather a problem of massive communication. Each excitatory neuron can connect to each dendrite of every single one of the remaining neurons (excitatory and inhibitory), in a single group of synapses. On the other hand, the inhibitory neurons can connect to each dendrite of every single one of the excitatory neurons. Each neuron can connect with a percentage of the other neurons. The main reasons that the network is not fully connected, is that the number of neurons cannot cover all the synapse groups. The fan-out of the neuron is given by:

$$Fanout_{neuron} = InterconnPercent \sum_{i=0}^{i=NumberOfNeurons-1} NumberOFDendrites$$

The interconnectivity is defined randomly, based on the interconnectivity rules, and each neuron is connected to all of the dendrites of the remaining neurons in a random group of synapses. We store the interconnection in off-chip memory to avoid the communication problem. Also, this is a great advantage of our design because the user can define the interconnection percentage and the interconnection probability distribution function.

3.4 Application Overview

The aim of this work was to develop a flexible simulator that biology scientists can use to simulate different biological experiments. Hence, we build a hardware-based simulator whose entire configuration is generic. All the network parameters are selected by the user and they are stored in the external memory of the hybrid platform. In the contexts of the diploma thesis [33], a “Real-Time visualization of neuroscience model simulation” has been developed, using the game engine Unity3d. The application is an easy to use interface to create the input simulation data and to visualize the results data based on my previous diploma thesis [27]. In this thesis,

we developed the interface in a way to be compatible with the new mechanisms and features. The result is a complete application that neuroscientists can use for their experiments.

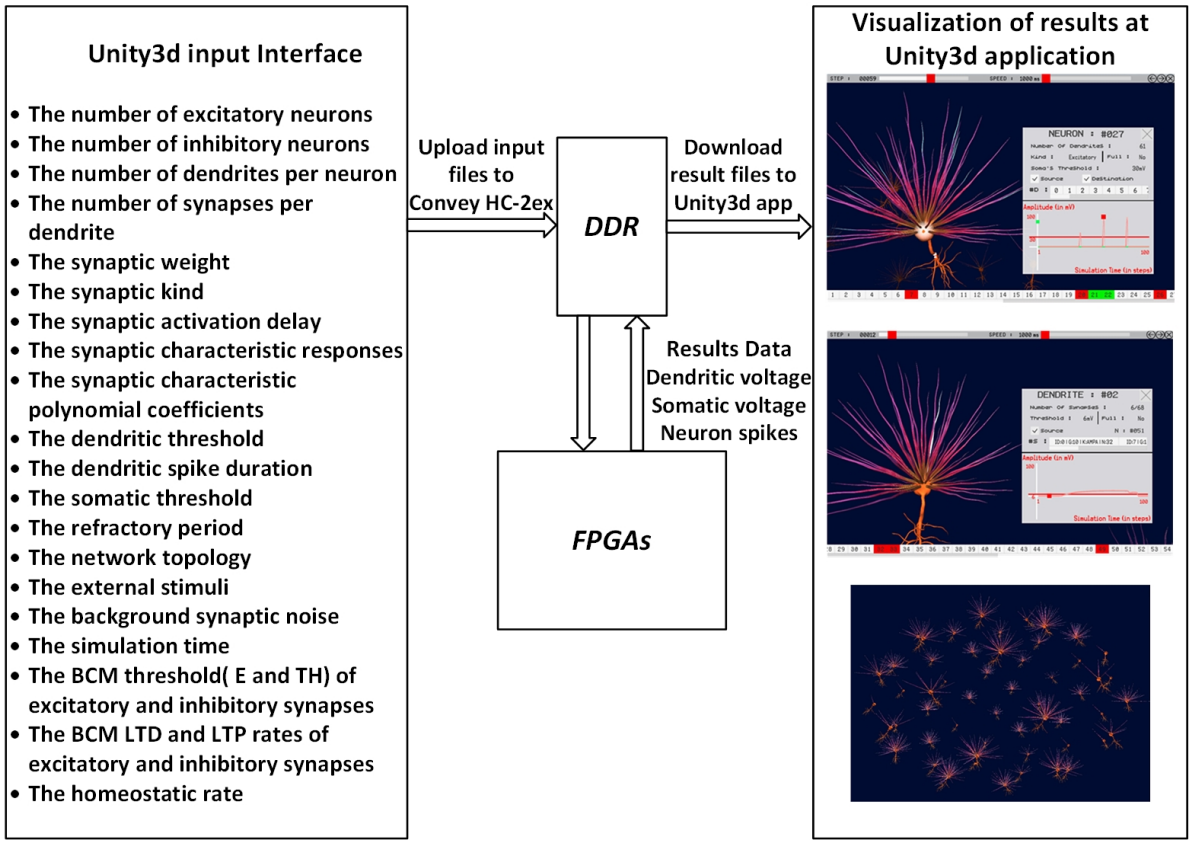

Fig. 3.10 General preview of the complete application

The figure 3.10 is a general preview of the application and shows the input parameters that the user can select from the application. The left side shows more specifically the input data. The users set the number of neurons, dendrites per neuron and synapses per dendrite without exceeding the limits of our design. In addition, they set the synaptic data (synaptic weight, type, delay), dendritic data (threshold, spike duration), soma data(threshold, refractory period), characteristic responses data (coefficients, discrete values), connectivity profile, external stimuli, homeostatic data(rate), BCM data(E/TH thresholds, rates) and of course the simulation time. The input files are generated based on the selected parameters and automatically upload to the Hybrid Platform Convey HC 2ex and store at the external memory of the platform. Researchers use random number of dendrites per neuron, number of synapses per dendrites, synaptic weights, dendritic/somatic thresholds and connectivity for performing different experiments. The input files are created to follow the same pattern. Chapter 4 explains in detail the different input data with the implementation of the architecture. The simulation runs (more details of

the FPGA architecture at the Chapter 4) at the four FPGAs of the platform and at the end of the simulation, the results (dendritic amplitudes, somatic amplitudes and neuron spikes) automatically are downloaded to the application and the 2D/3D visualization begins.

3.5 System dimensioning

The development of the biologically inspired neural network simulator, was a challenge because of the many different data that should be processed and the complexity of the detailed problem. Although it would be ideal if all the data could fit in internal FPGA memory (BRAM), the total number of simulation data are many orders on magnitude larger capacity than the number that can be supported on-chip and so, our architecture had to exploit external memory to be able to store them. The FPGA architecture was implemented on the Convey HC-2ex hybrid FPGA super-computer, because of the large amount of data that should be stored in external memory and the required memory bandwidth. The Convey HC-2ex has four Virtex-6 LX760 FPGAs and all FPGAs share the same memory and a coprocessor. The platform has 14 memory controllers for parallel read and store. The main problem was the distribution of the data. Hence, we had to decide which data will be stored in the internal and in the external memory. This significant decision was critical because it limits our further options. We decided to store in the internal memory of the FPGA all the input data except for neuron interconnectivity and synaptic noise. The frequent use of this information led us to this approach, so that the system takes advantage of the high bandwidth of internal memory at random data. By contrast, the external memory of the platform offers high bandwidth in burst memory accesses, so we only stored the neuron interconnectivity and background synaptic noise in the external memory and read them in burst mode when it's necessary, i.e. one complete pass per iteration. The disadvantage of this approach is the limitation to the number of neurons, dendrites and synapses that could be simulated.

Two decimal accuracy for the voltage is more than adequate resolution given that the biological-relevant features of neuronal responses are of the order of tens of voltage (e.g. spike amplitude is of the order of 100mV). Floating point arithmetic allowed us to have more than adequate accuracy, whereas the required logic was not the critical resource in the implementation (the on-FPGA BRAM memory was, as will be shown). In addition, we used two's complement arithmetic, because of the inhibitory synapses. An important element of modeling is the simulation step of the biological model. Given that the highest time resolution for biologically relevant features of neuronal response is of the order of 1ms (Action potential duration: 2-4ms), we considered that 0.1ms as a satisfactory step for such simulations. So,

Table 3.2 Number of neurons, dendrites and synapses that the system can simulate

Number of neurons	Number of dendrites per neuron	Number of synapses per dendrite
≈ 240	64	128(groups)
≈ 15120	1	128(groups)
≈ 7440	32	128(groups)
≈ 480	64	32(groups)

we sampled per 0.1ms the characteristic responses of all the kinds of synapses and stored the discrete values on chip memory.

Another problem was the modeling of the big number of data that is stored in the internal memory. Each synapse requires 36bits for storing its information. More specifically, the types of synapse is four, (AMPA, NMDA, GABAb, GABA), so 3 bits is a fine number for the coding of different types. Another aspect of the synapse is the counter for the sampling of the response curves. The NMDA has the longest curve, lasting 350ms, so the sampling counter is coded with 11 bits. We define the same width for all the type of synapse because of the flexibility of the simulator. An important element of the synapse is the synaptic weight, which are encoded with 14 bits. The synapse activation delay is coded with 7 bits and 1 bit is necessary to characterize the activation of the synapse. The values of the characteristic response of any type of synapse, the dendritic and somatic threshold, the synaptic and somatic voltage are coded with 16 bits due to the two decimal digit accuracy.

The limitation factor of our system is the internal memory after the storage of the synaptic data and the characteristic responses to on-chip memory. The synaptic information and the synaptic curves are stored in the Block Rams, determining the size of the network. The Convey platform has four Virtex-6 LX760 FPGA, each of them has 25.920 Kbits BRAM. Table 3.2 analyzes the total number of neurons, synapses and dendrites that the system is able to simulate, considering the amount of information to be stored in Block Rams and the available resources. The dimensioning is done based on the number of synapses and the characteristic responses, which can be stored in internal memory. In addition, we took into consideration that the memory controller of the Convey platform use about 13%-15% of the total BRAM. The total number of synapses per neuron is the number of memory locations to be committed. Each memory location holds a synaptic information. It is obvious that the number of neurons are increases while the number of dendrites and synapse decreases.

Table 3.3 also shows the total number of the neurons, dendrites and synapses after the curve fitting optimization. Using this technique we release some Block Rams because the characteristic responses aren't stored.

Table 3.3 Number of neurons, dendrites and synapses that is predicted can simulate the system with curve fitting technique

Number of neurons	Number of dendrites per neuron	Number of synapses per dendrite
≈ 255	64	128(groups)
≈ 16065	1	128(groups)
≈ 7905	32	128(groups)
≈ 510	64	32(groups)

3.6 System functions

The above analysis of the neurons, dendrites and synapses led to the division of the system into five basic functions: the initialization of the system, the update of synapse state, the computation of neuron state, the learning state and the data storage.

The initialization is only performed at the beginning of each simulation and includes the data that the user have set and they have been generated from the software interface. More specifically, the initialization of synaptic data, dendritic data, somatic data, characteristic synaptic responses and parameters of the system are the sub-functions that are performed during the initialization state.

The update of synapse state is iterated in each simulation time step. During this process, the interconnectivity of the neurons, that is stored in the external memory, and spike vector are combined in order to activate the synapses. The synaptic noise is also placed in the external memory because it follows the same logic, as we will see in the next chapter. Hence, the system doesn't need to store the topology of the neurons, saving system resources.

During the computation neuron state, the action potential is calculated for a simulation step. The calculation becomes in dendritic and somatic layer, as we analyzed in the previous chapter. The synapses produce an electrical output, according to their response counter, synaptic type, synaptic weight and status. At the end of the calculation, the vector with the neuron spikes, is updated. The learning state is also performed during the computation state, and changes the synaptic weight based on plasticity mechanisms. A simulation step is completed when the somatic voltage, dendritic voltage and neuron spike are stored in the external memory(the data storage state).

Chapter 4

Design and implementation of the system architecture

This chapter analyzes the design and implementation of the system architecture in reconfigurable logic. Both the design and the implementation of the architecture were taking place at the same time, in order to be in close consultation with the biologists. We will present the basic differences with our previous and explain the problems and the difficulties that we have faced. In addition, the detailed description of the input/output data is referred.

In the diploma thesis [27], a system that could simulate a network with up to 70 neurons, 64 dendrites per neuron and 512 synapses per dendrite, was built. The simulation step had defined at 1ms. This spiking neural network can model some basic neural mechanisms. The system was at an earliest stage because it was our first try to implement neuromorphic algorithms in hardware, so the simulator faced many problems like:

- many bugs in the biological level
- few parameters were as input in the system
- the advantages of the hybrid platform were not exploited in the performance level
- poor biological mechanisms

In this thesis, we designed the architecture from the scratch because the new features of the simulator, in biological and performance level, demand a different architecture approach. Of course, we kept the same primary design philosophy and the positive elements of it. We did lots of optimizations and we implemented some new mechanisms from the biological aspect, in order to improve the biological accuracy of the simulator.

Figure 4.1 presents an abstract picture of the neural network architecture. Considering that each FPGA can simulate up to 60 neurons and the platform has 16 MCs, we group the

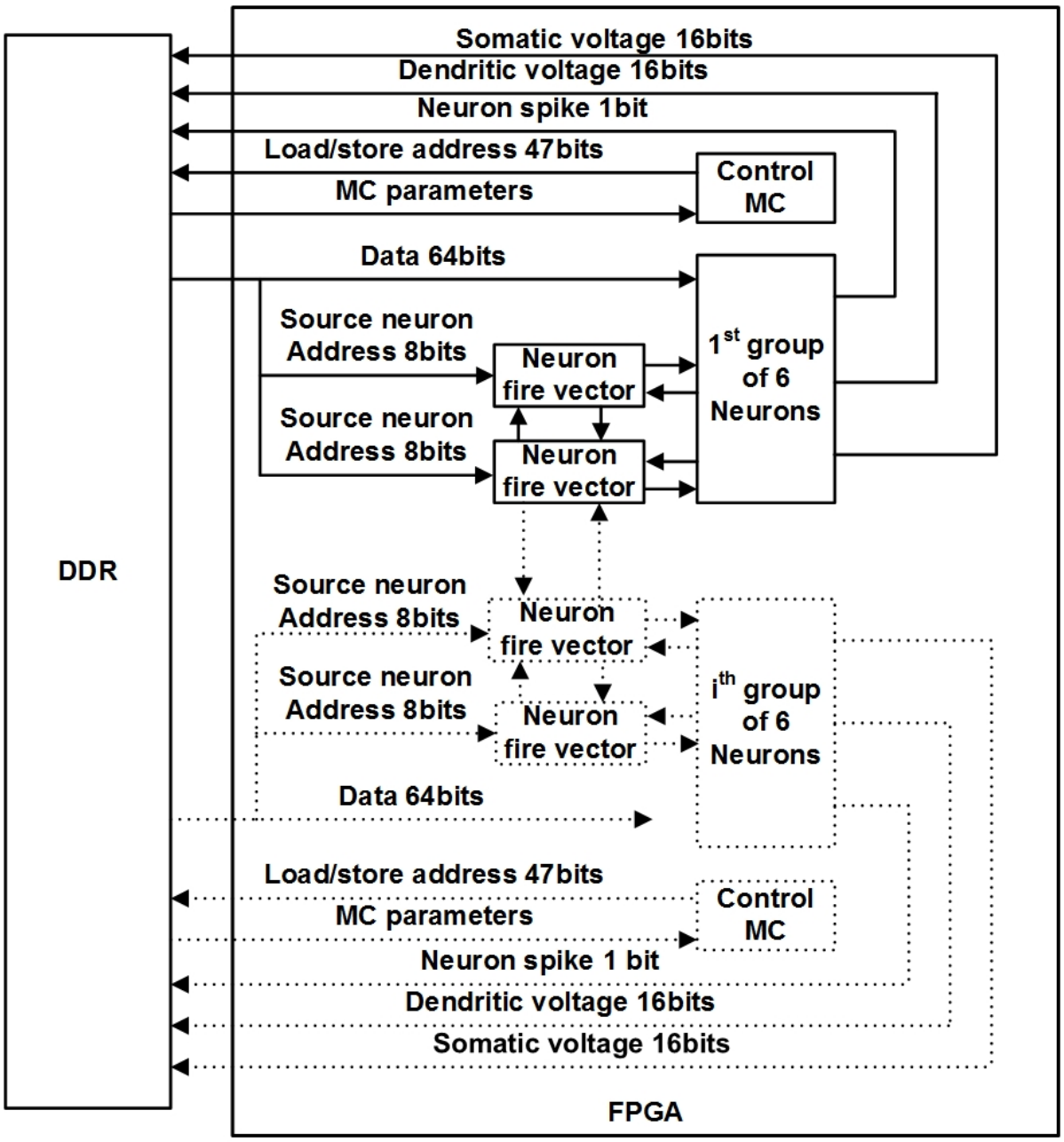

Fig. 4.1 Abstract neural network architecture diagram and input/output data from the external memory

neurons per 6 and each group is connected with one memory controller of the platform in order to load/store data in parallel. It is understandable that any smaller group of neurons (e.g. per 4) could not fit with our design not only because of the memory controller limit but also because we execute additional functions in parallel, as we will see in section 4.9. Each group has a control unit that manages its memory controller interface and two identical vectors

that keeps the spikes of the neurons. These vectors are very important because they allow the parallel update of the synapses at two different stages, as we will see in the synaptic update state (section 4.2). Every neuron of each group receives input data from the external memory in parallel, but only one neuron holds them because of a unique neuron id. The data that are read from the external memory of the platform have the maximum size of 64bits and are loaded via the host of the Convey platform. The 64bits quantity offers a satisfactory size for information transfer. In this architecture, the datapath of 64bits is divided into two smaller categories, of 32bits each, which are used as addresses and/or as data. Each function uses all or a portion of the available bits of each 32bits field, depending on its requirements. Figures 4.2 4.3 4.4 show how the form of the 64bits is associated with different parts of the datapath.

A general difference of the new architecture is the grouping of neurons that takes advantage of the parallel memory controllers of the platform. Each group of neurons is connected with one Memory Controller in contrast with our previous work where all the neurons were connected with a unique MC. Each MC is responsible for a small number of neurons, so it can read/write only a specific space of the memory that is defined in the initialization of the system. In the setup of the system, each control Memory Controller Unit is initialized with the parameters, including the read/store address that is responsible. The MC exploitation is significant extension, considering that the update and store state perform each simulation step. However, the management of all the MCs and their synchronization is a difficult part of the design.

In the following sections, we will analyze the system architecture in the aspect of a neuron and we will show the architecture of five different functions. We will analyze in depth the form of data, that the functions receive from the external memory, as well as their distribution in the system. Then, we will explain the overall network architecture, arithmetic units, integration in the four FPGAs and problems that arose.

4.1 Initialization of the system architecture

Figure 4.2 represents a more detailed aspect of the initialization architecture of the system. The initialization is only performed at the beginning of each simulation and includes the initialization of synaptic data, dendritic data, somatic data, characteristic synaptic responses or polynomial coefficients and parameters of the system. Figure 4.2 also shows the data form of each function. Following a general agreement, the 32 MSB bits is the address of the neuron, dendrite and/or synapse. Of course, some functions have custom form depending on the requirements. The useful data that are read from DDR differ, depending on the initialization state.

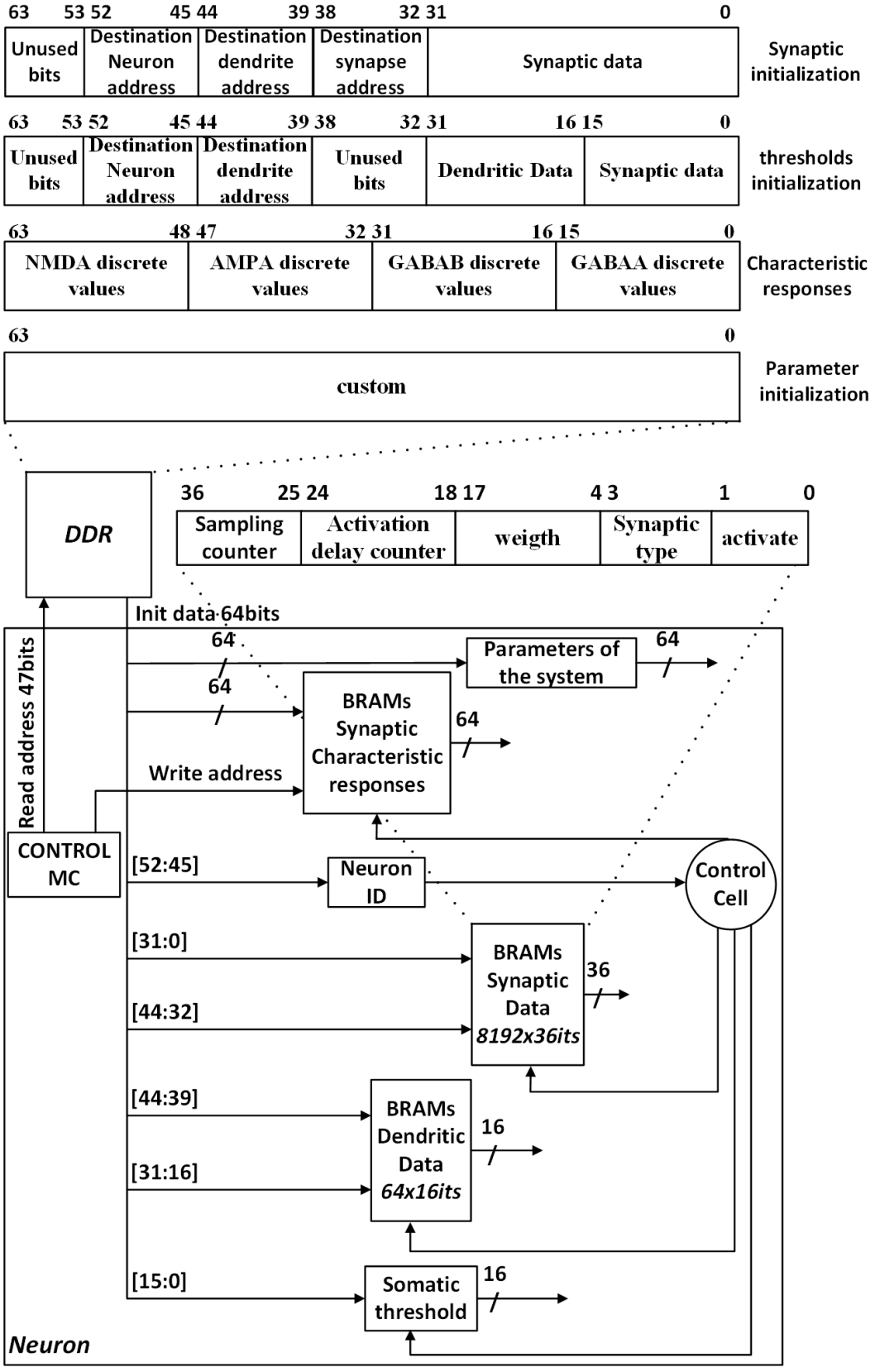

Fig. 4.2 Architecture of the initialization of the system and the form of data

Each neuron has a dual port Block RAM (BRAM) of 8192 lines depth and 36bits width, plus a distributed single port BRAM of 64 lines depth and 16bits width that can keep all the synaptic and dendritic data of the neuron, respectively. The synapses are initialized with the parameters that characterize them such as the type of synapse, the synaptic weight, the activation counter, the sampling counter and synapse activation state. Each neuron also has a unique id that indicates its address. Each block of 36bits width keeps the data of one synapse. The distribution of the data is also shown in the Figure 4.2. The initialization is a simple function, the data is read from the external memory and each neuron keeps their data based on the unique neuron id. The destination neuron address indicates the neuron that the data aims. The comparison between destination neuron address and neuron id is made to decide if the synaptic data belongs to the neuron. The destination dendrite and synapse address are the write address where the data are stored. The control unit is responsible to control the appropriate write enable signals of the BRAMs.

The dendrites and soma are initialized with the dendritic and somatic threshold respectively, on the assumption that if a synapse, a dendrite or a neuron does not take positive value, it remains inactive and does not affect the system. The initialization is done based on the same way as previous. The threshold initialization needs fewer bits to address the data because the dendrites are in upper level of the neuron's hierarchy than the synapses. The neuron and dendritic destination address are the write address in the register and BRAM, respectively. Of course, the synapse address is not required, as the form of data indicates(Figure 4.2).

The characteristic synaptic responses are stored in four additional single port BRAMs that the neurons also have. Each synaptic curve has different duration, so the depth of each BRAM differs from the others. The AMPA and $GABA_a$ responses have 512 memory locations, the NMDA 2048 memory locations and $GABA_b$ 1024 memory locations but all the BRAMs have 16 bits width. The coefficients replace the characteristic synaptic values, in case the synaptic curves are modeled with polynomial functions. The coefficients values are stored in registers. The parameters of the system (BCM excitatory/inhibitory thresholds and rates, synaptic activation delay, dendritic spike duration, refractory period, simulation time, external stimuli, MC addresses) are also stored in registers. The 64bits quantity fits exactly in the 4 discrete values of each synaptic type (Figure 4.2), so the design only requires to create the write address to the BRAM. As we mentioned above, the BRAMs don't haven the same depth because of the different duration of each synaptic type. In that case, the discrete values of the shortest synaptic type and the write enable of the specific BRAM are set to zero.

The storage of the dendritic threshold data in distributed BRAM was the basic improvement of our previous work. The storage of the data in BRAMs was unnecessary and waste of resources, due to the small quantity of the dendritic data. A new feature of the simulator is

the plurality of the data inputs of the system, such as the characteristic synaptic responses, plasticity parameters, synaptic parameters and dendritic parameters. Hence, the simulator can run different experiments based on the user preferences. A design decision that affected the total system, was the modeling of the synaptic activation delay mechanism. The synaptic activation delay is incorporated into the BRAM that keeps the synaptic parameters, as a result the width of the BRAM is increased. In addition, the simulation time step is defined at 0.1ms in contrast with the previous work that was at 1ms. Hence, the characteristic responses had to be sampled with the new time step, so the total number of the discrete values of each synaptic type was larger than the previous design occupying more BRAMs of the system. The above reasons constrained the available resources of the system, as a result the total number of neurons that the system can simulate, were decreased in 60 neurons per FPGA.

4.2 Update of synapse state architecture

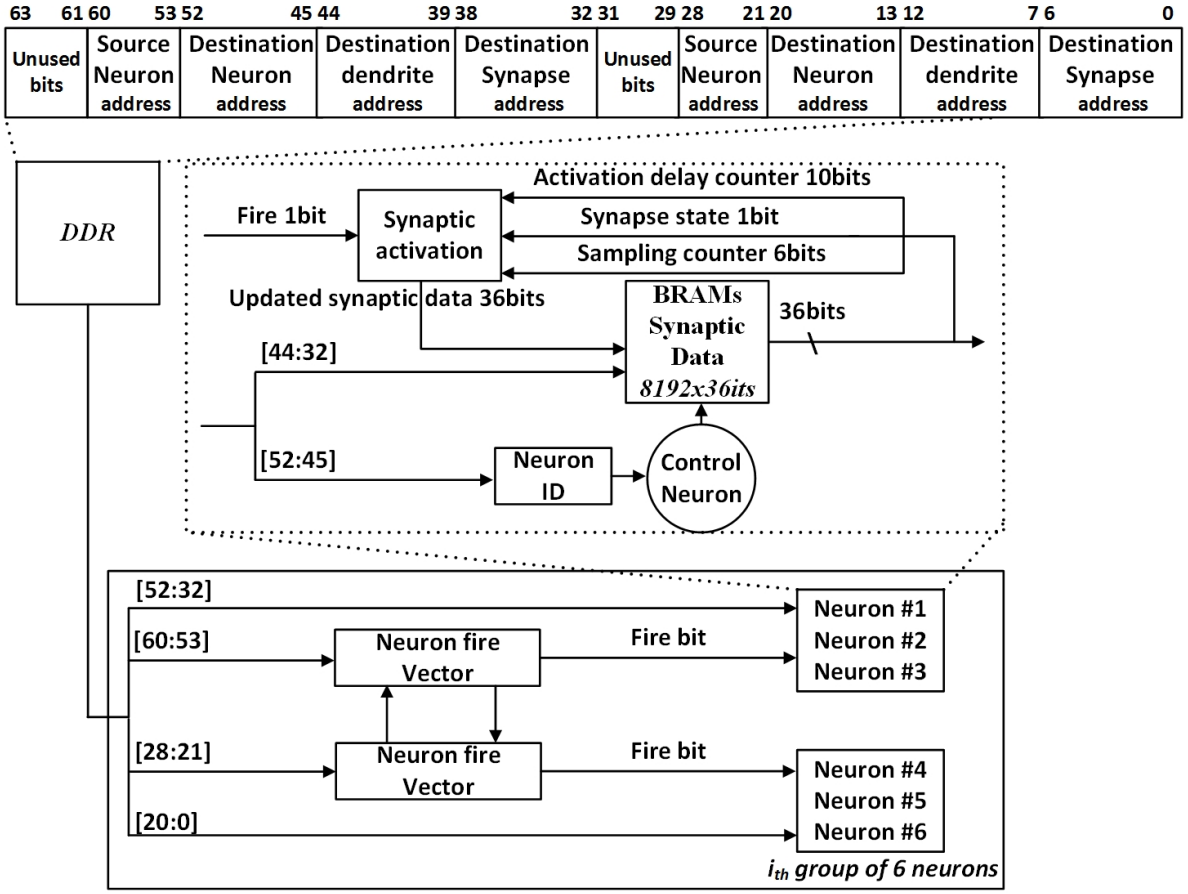

Fig. 4.3 Update of synapse state architecture and the form of data

The update of synapse state is iterated in each simulation time step. During this function, the synapses are activated, unless the neuron which is connected with them has triggered a spike. Figure 4.3 shows the architecture of the update function and the form of the data that read from the external memory.

Algorithm 1 update synaptic state

```
1:  procedure SYNAPSE ACTIVATION
2:      synapseActivDelay ← constant value of the Synaptic activation delay
3:      synapseState ← Synapse state (active or not)
4:      synapseActivCnt ← Synapse activation delay counter
5:      synapseSamplingCounter ← Synapse sampling counter
6:      NeuronFire ← spike from neuron which is connected with the synapse
7:
8:      if synapseActivCnt ≠ backgroundSynapseTag then
9:
10:         if NeuronFire = 1 and synapseActivCnt = 0 then
11:             synapseActivCnt ← synapseActivCnt + 1
12:             synapseState ← 0
13:
14:         else if synapseActivCnt > 0 and synapseActivCnt < synapseActivDelay then
15:             synapseActivCnt ← synapseActivCnt + 1
16:             synapseState ← 0
17:
18:         else if synapseActivCnt = synapseActivDelay then
19:             synapseActivCnt ← 0
20:             synapseSamplingCounter ← 0
21:             synapseState ← 1
22:
23:         else if synapseActivCnt = 0 then
24:             synapseActivCnt ← 0
25:             synapseState ← 0
26:
27:         end if
28:     else
29:         if NeuronFire = 1 then
30:             synapseState ← 1
31:         end if
32:     end if
33: end procedure
```

The update data,is stored in external memory and constitutes the neuron interconnectivity-network topology. The system has neuron fire vectors 240bits that are only stored the spikes

of all the neurons. In this way, the system doesn't need to know the neuron topology,saving system resources. The neuron interconnection topology is read in burst mode from the external memory. The source neuron address indicates the neuron that the synapse is connected and is input at the neuron fire vector. At the same time, the destination neuron, dendrite, synaptic address read from the BRAM the specific synapse which will be updated, following the same logic as the initialization state (i.e. neuron id et al.). The activation of synapse depends on both the neuron spike and an additional mechanism that is analyzed more specific in the algorithm 1. The activation state, the activation delay counter and the sampling counter constitute important factors of the activation mechanism.

The structure of the input data follows the same logic as the initialization function for the exploitation of shared resources. However, the design takes advantage of the 64bits and update in parallel two different synapses, offering an addition parallel stage in our architecture. As we have mentioned, the neurons are grouped per six, offering parallel read and write. Figure 4.3 also shows the architecture of the parallel update. Each group of neurons have two 240bits vectors with the neuron spikes, offering two parallel update machines.

The algorithm of the synaptic activation is changed in compare with our previous work. The synaptic delay mechanism confused the procedure and the overall architecture. In addition, the parallel synaptic update is a new element of our design, giving performance optimization(up to 2x speedup).

4.3 Synaptic noise state architecture

At the same time with the update of the synaptic state, the mechanism of the synaptic noise state is taking place. The synaptic noise activates some synapses-background synapses- during the simulation time, based on Poisson distribution. This procedure is independent from the neurons state,as the algorithm 1 shows. The total number of the synaptic noise data is calculated at the setup of the system according to algorithm 2. The procedure, for generating synaptic noise data, is based on the fact that the estimated probability of activating a background-synapse during a short interval of duration **dt** is **fr*dt**.The **fr** is the frequency of the synaptic noise activation and **dt** is the simulation time step. The algorithm progresses, for all the synapses, through the simulation time in small steps of size dt and generates, at each time step, a random number **x** chosen uniformly in the range between 0 and 1. If **x<fr*dt** at that time step, the synapse is activated, otherwise it is not. The total amount of synaptic noise data depends on both, frequency and simulation time. Hence,in some cases, the system may not be able to keep the synaptic noise data in on-chip memory. Considering that the system must be flexible and the total amount of the synaptic noise data is unknown, we decided to store them in the external

memory of the system with a handy way, placing them sorted. The sorting is taking place based on the activation time.

Figure 4.4 shows the form of data for this function. The 32 MSB are the synaptic address and the 32 LSB are the time of activation. An additional MC of the platform is responsible to bring the synaptic noise data from DDR and only keeps the proper ones. The filter is the simulation time, so a comparison of the current simulation time step and the activation time of the synapse is made. An important note is that the system must know which synapses are background-synapses.Hence, during the setup of the system, the activation delay counter, of the background synapses, is set with a specific constant value -127- which is the tag of the background synapses. The background synapses don't follow the activation algorithm of the regular synapses but they are activated immediately, as the algorithm 1 shows. In case the comparison is false, the system stops loading synaptic data and defines the new load address. In contrast to the synaptic update function, the synaptic noise state doesn't read all the data from DDR but only the ones it needs, saving simulation time.

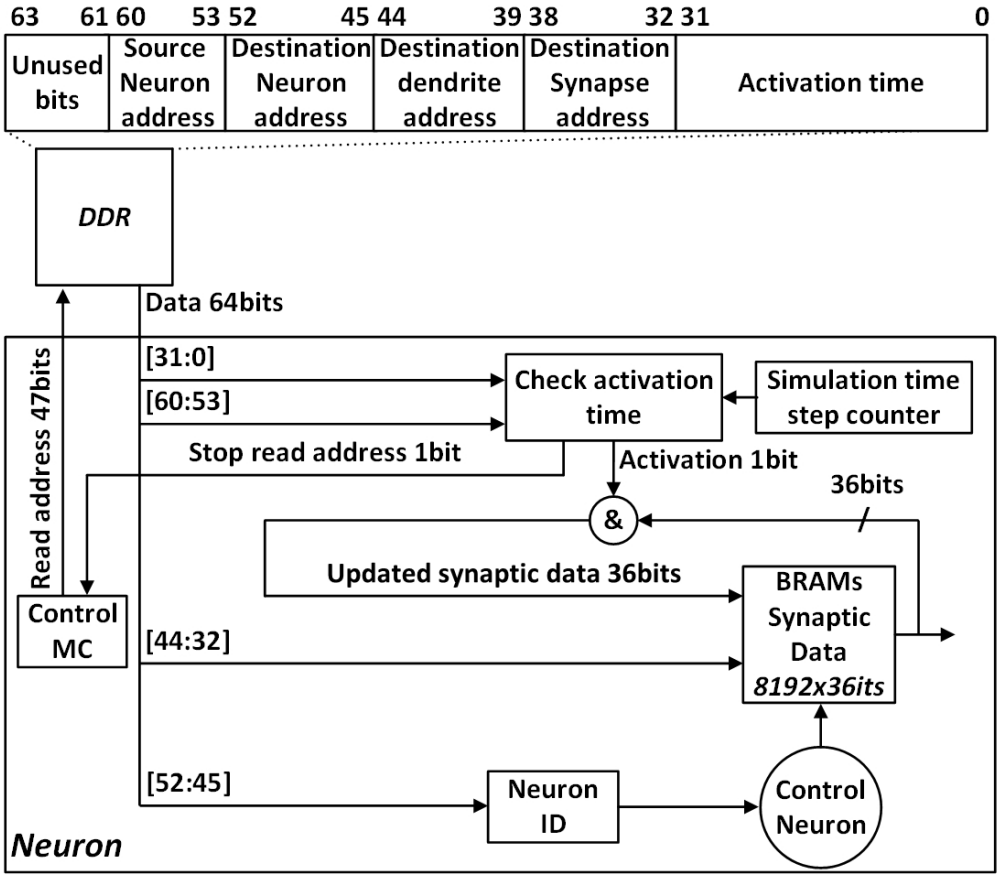

Fig. 4.4 Synaptic noise state architecture and the form of data

Algorithm 2 background synaptic noise

```
1:  procedure BACKGROUND SYNAPTIC ACTIVATION
2:
3:      dt ← simulation time step ▷ Our design has dt=0,1ms
4:
5:      for each synapse in unconnected synapses do
6:          for each t in simulationTime do
7:              x = random[0,1]
8:              if x < Frequency * dt then
9:                  SynapseState = 1
10:             else
11:                 SynapseState = 0
12:             end if
13:         end for
14:     end for
15: end procedure
```

4.4 Computation state architecture

The computation neuron state (Figure 4.5) is performed to calculate the action potential in each simulation time step. The system doesn't receive data from the external memory since all the needed data are in the BRAMs and in the registers. Each neuron has a 64 depth and 16 width distributed BRAM whiere the dendritic voltage is stored At the beggining of the calculation, all neurons are evaluated in parallel, and they serially read the synaptic data from their BRAMs. The synaptic activation module is responsible to check the synaptic state based on the synaptic activation counter delay and to increase the sampling counter. The updated data is stored in the BRAM after performing the plasticity rules (more details in section 4.5). The design is fully pipelined, hence the synchronization was a complicated procedure after the addition of the learning mechanisms. The synaptic activation signal and the sampling counter indicate when a synapse is activated. The sampling counter is an address in the BRAMs,where the characteristic synaptic responses are stored, and it samples the discrete values of the curves. A multiplexer selects the useful discrete value based on the type of the synapse.

According to the biological model, the system operates as follows. The discrete values in hardware are multiplied with the weight of the synapse and the result is summed in the dendritic layer. When all the synapses of the dendrite are computed, the total voltage of dendrite is compared with the dendritic threshold. If this value is greater than the threshold, it will be transmitted to the somatic layer. At the same time, the value will be stored in the dendritic voltage BRAM. When the dendritic amplitude will become less than the threshold, the stored value will be exponentially decreased and will be transmitted to the soma. The exponential

calculation begins at the start of each dendrite's computation, so it doesn't affect the overall time of computation. The exponential factors, which are input to the system, are multiplied with the dendritic voltage and the new value is stored in the BRAM. The exponential factor has been calculated to decrease the voltage based on the dendritic spike duration, so the system doesn't need additional counter to calculate the dendritic spike duration. Immediately upon all dendrites are computed, the soma checks if the overall voltage has exceeded the soma threshold. The neuron triggers a fire when the refractory period is over, so each neuron has its own timer to count this period. The computation of the neurons is fully pipelined. When the process is finished, the array with the neuron firings is updated.

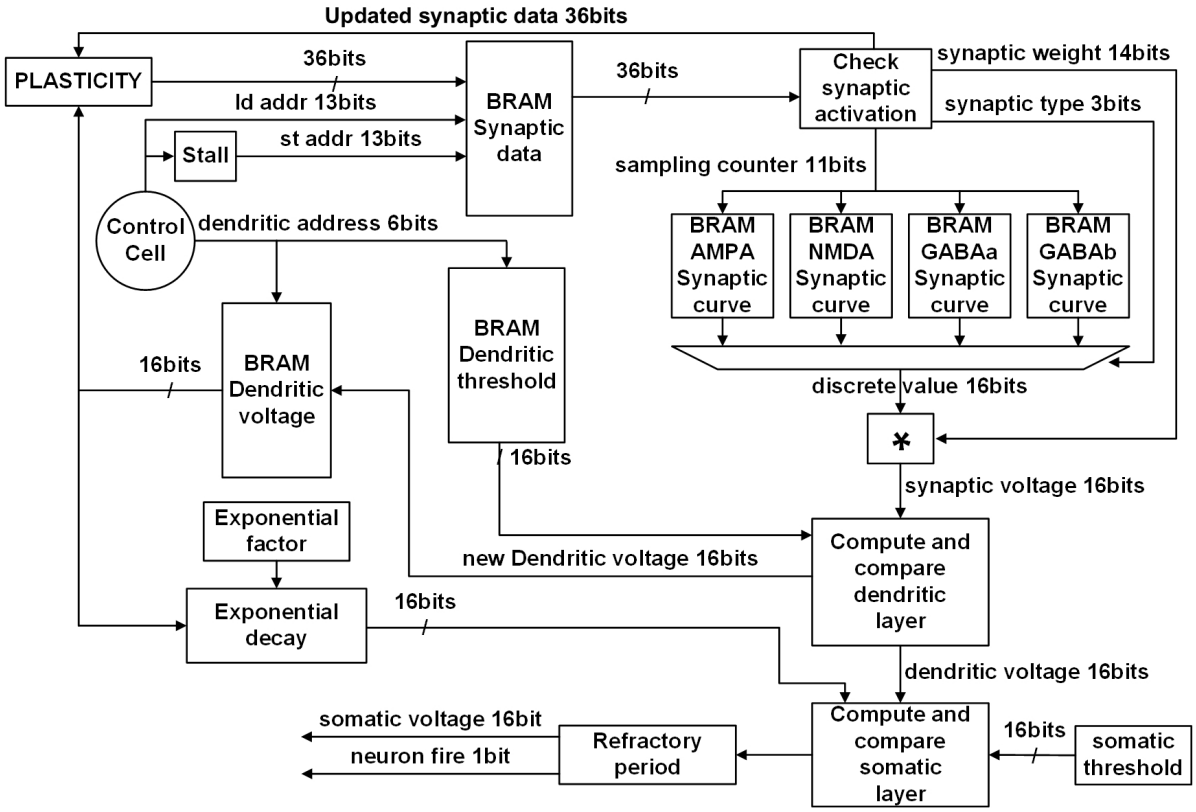

Fig. 4.5 Compute state architecture

4.4.1 Different approach of the characteristic synaptic responses

As we discuss in the chapter 3, we approached the synaptic characteristic responses from a different aspect. The characteristic responses were studied with a curve fitting technique and were analyzed into mathematical functions. At the initialization state, all the coefficients (a, b, c) of the equations are stored in registers. The initialization data is from the external memory,so the user can define the coefficients of the equations on the basis of its own experimental data.

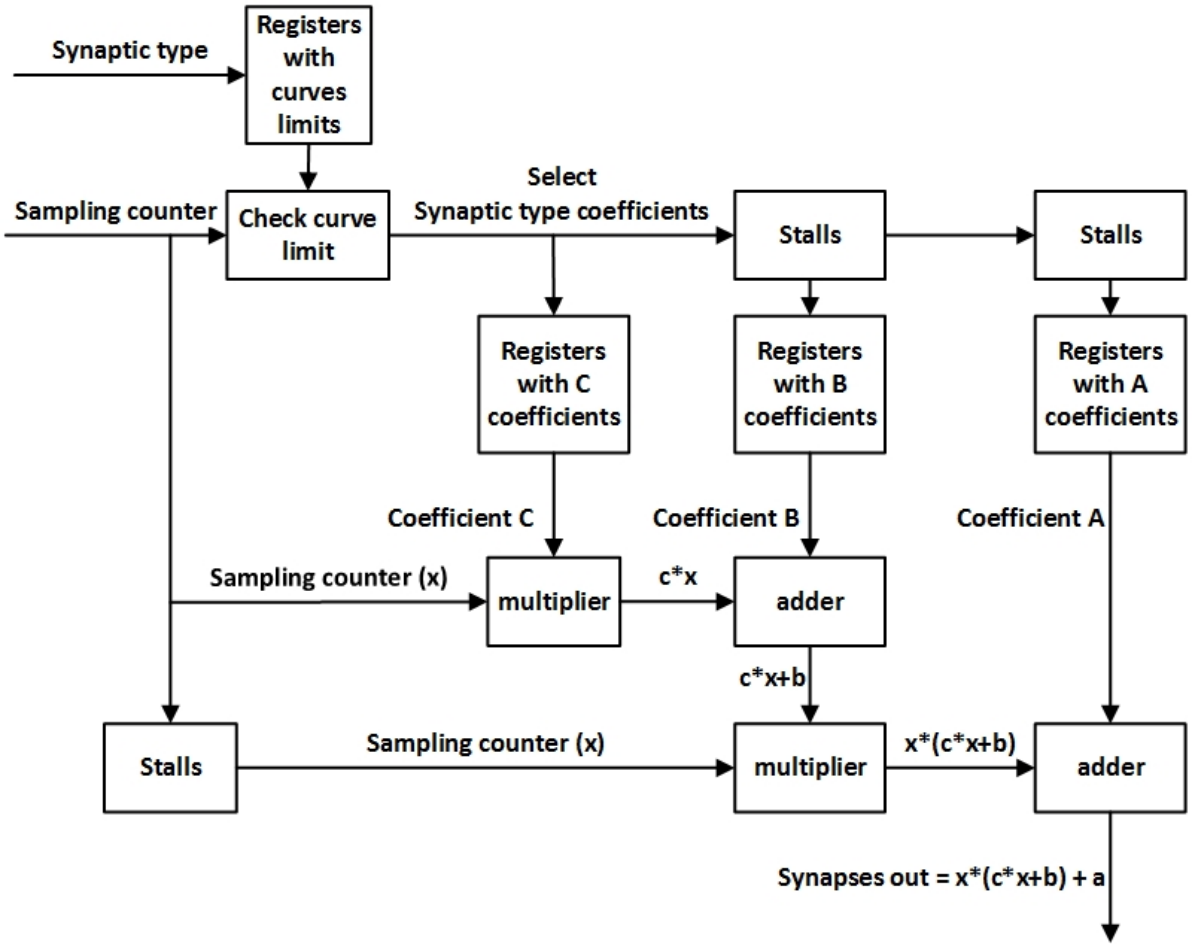

Fig. 4.6 Polynomial architecture

We decided to use fixed point arithmetic because of the limitation of the available DSPs.The BRAMs with the characteristic responses are replaced with a module that calculates the synaptic electrical output. Figure 4.6 shows the polynomial module, that has as input the sampling counter (denoted as x value in the equations) and the synaptic type. This combination defines the parameters of the function. The module has two multipliers and three adders that "build" the equation. The module is pipelined in order to fit with the existing architecture. For that reason, the sampling counter and the synaptic type are passed through the pipeline registers, creating

a deep pipeline. The latency is 15 clock cycles. The new modeling of the characteristics synapses saves three 36K BRAMs, in each neuron of the network, and totally 180 36K BRAMs. Considering that each neuron needs 8 36K BRAMs for storing synapses, the FPGA can ideally simulate approximately 15 more neurons.

4.5 Learning state architecture

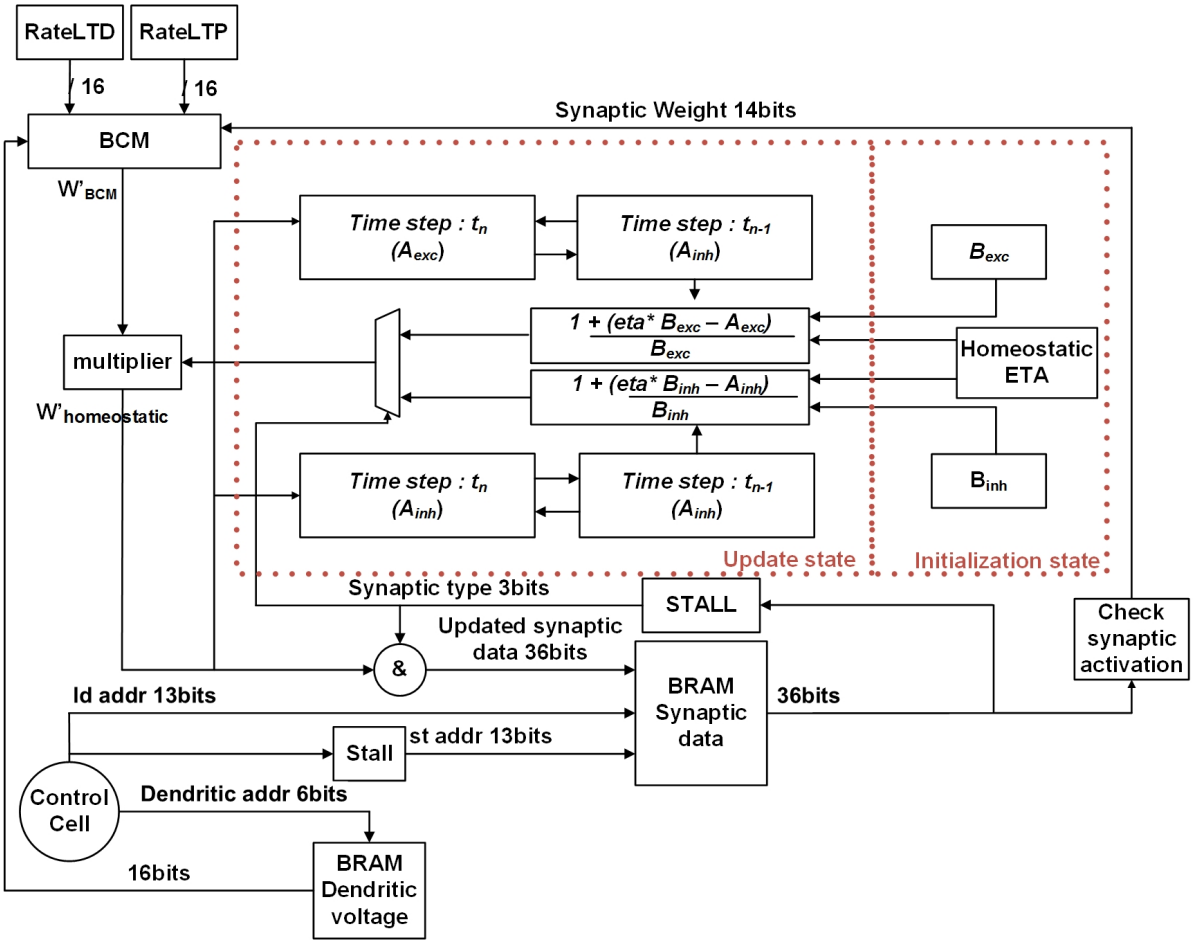

Fig. 4.7 BCM rule and Homeostatic plasticity architecture

The learning state is also performed in each simulation step. We designed an architecture that is executed in parallel with the previous functions, calculating the needed values. Hence the learning mechanism does not influence the total computation time, adding only a few cycles in the total latency. Figure 4.7 represents the architecture of the learning state and the parts of the design that are calculated in the different states.

In particular, at the initialization state the system sums the total excitatory and inhibitory synaptic weight of each neuron and keeps the values in registers. The background synapses do not participate in the total weight. The calculated values are the B_{inh} and B_{exc} of the equations 2.2 2.3 and do not change their values during the simulation. At the computation state, the system exploits the serial access of the synaptic BRAM, so the BCM rule can also be executed. If the synapse is activated, the weight must be increased or decreased,depending on the level of the dendritic voltage, according to the equations 2.2 2.3. The new synaptic weight that has been calculated from the BCM rule is multiplied with the quotient of the equation that has been calculated during the update state. So, the multiplication can be done immediately, and doesn't influence the pipeline logic of the architecture. We are able to calculate the quotient from the B_{inh} and B_{exc} which are the total weight of the excitatory and inhibitory synapses in the previous time step. The new synaptic weights that are calculated from the homeostatic rule, are combined with the other synaptic data and the updated values are stored in the BRAM. The updated weights are summed and the new summation constitutes the new total weight of the excitatory/inhibitory synapses of the neuron. The synaptic data and the write address must be stalled, until the new weight is calculated. However, the rest of the system continues to operate independently and calculates the total voltage in dendritic and somatic layer. Hence, the synchronization was the difficult part of the learning state architecture because it had to be integrated harmoniously with the whole system.

4.6 Store state architecture

Figure 4.1 also shows the outputs of the neuron. In the function of the computation, each of the neurons have their state calculated in parallel and, at the end, a vector that keeps the neuron spikes, a module that stores the somatic voltage and an internal memory that stores the dendritic voltage, is updated and stored in the external memory. The data constitutes a large amount of information, so the storage is done in parallel way, at the external memory. The neurons are grouped in 6 groups and feed a different memory controller of the convey platform.

4.7 Control Unit in the neuron level

Each neuron has a control unit that organizes and controls the functions in order to synchronize the system. The unit communicates with the general control unit, which informs what function will be executed, and the neuron id unit. During the initialization state, the control unit is responsible for reading and writing data in the memory, and in case of calculation, it reads and writes data sequentially in memory, with the help of a counter.

4.8 FPGA interconnection

The system is integrated in four FPGAs and each FPGA simulates different neurons. Each FPGA has access in specifics blocks of the external memory to read and write data, as the Figure 4.8 shows. The communication of the FPGAs is taking place through the DDR in a simple way. The only information that each FPGA must have, is the vector with the neuron spikes after each simulation step. Therefore, at the end of each computation state, each FPGA stores its neuron spikes (each FPGA simulates 60 neurons) with a store id in the remaining MSB bits, at a specific address of the DDR. The store id indicates that the values are not garbages but useful data. Then, each FPGA reads the address where the others FPGAs have stored their neuron spikes. Each FPGA reads the same address until the store id is correct and updates the vector with all the neuron spikes. At the update function of the simulation, one MC is responsible to clear the values with the neuron spikes in DDR, avoiding synchronization conflicts.

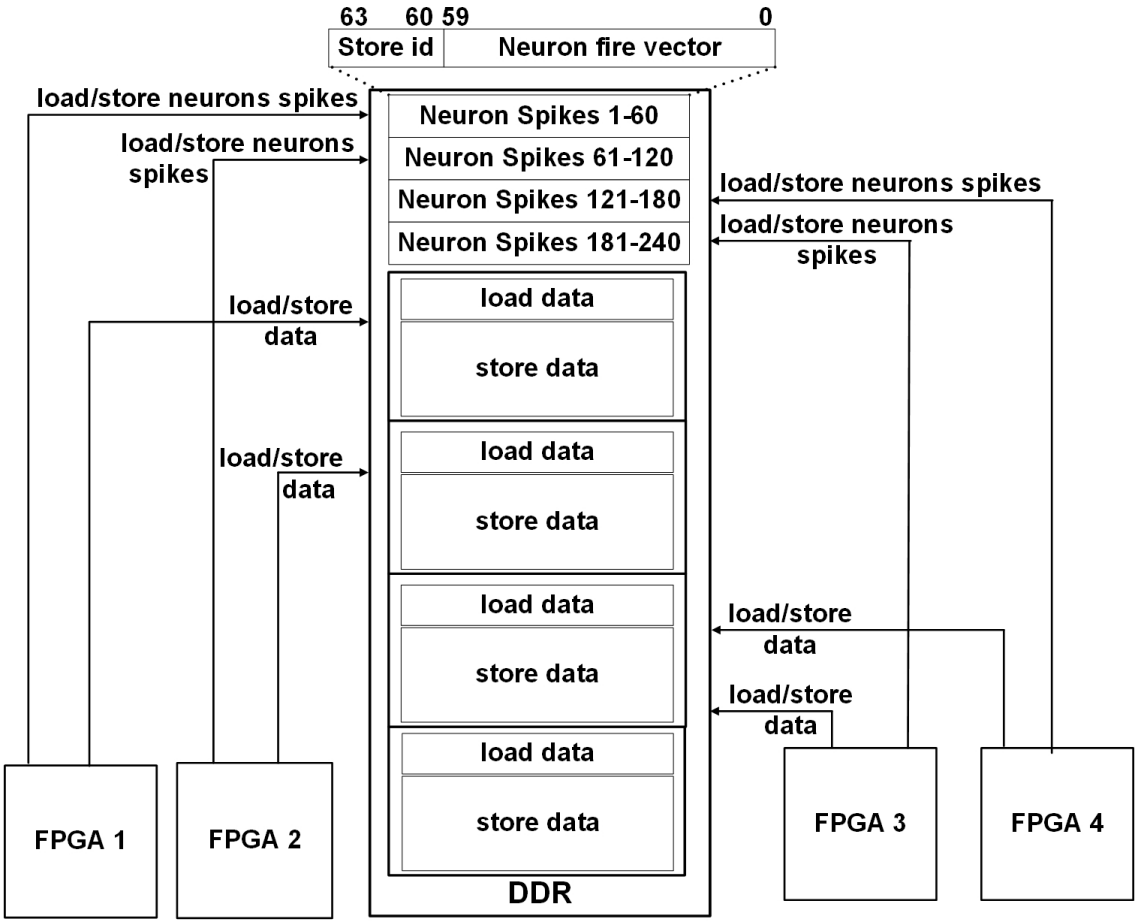

Fig. 4.8 Interconnection between FPGAs and data distribution

4.9 System flow

Figure 4.9 shows the flowchart of the system functions, including the parallel execution of them. The system initialization and the summation of the total excitatory and inhibitory synaptic weights, take place once during the system installation. The remaining functions are executed in each time step and constitute one simulation time step, 0,1ms. The synaptic noise, the update of synaptic state, the calculation of quotient for the learning state and the clearance of DDR are executed in parallel, exploiting all the available memory controllers of the platform and the independency of the data. In addition, the computation state and the learning state, as the Figure 4.9 shows, take place at the same time. The storage of the somatic voltage, dendritic voltage, neuron spikes and the update of the neuron spikes occur at the end of each time step. The network has a general control unit that takes as input value the desired simulation time. The general control unit is responsible for iterating the functions until the time steps finish.

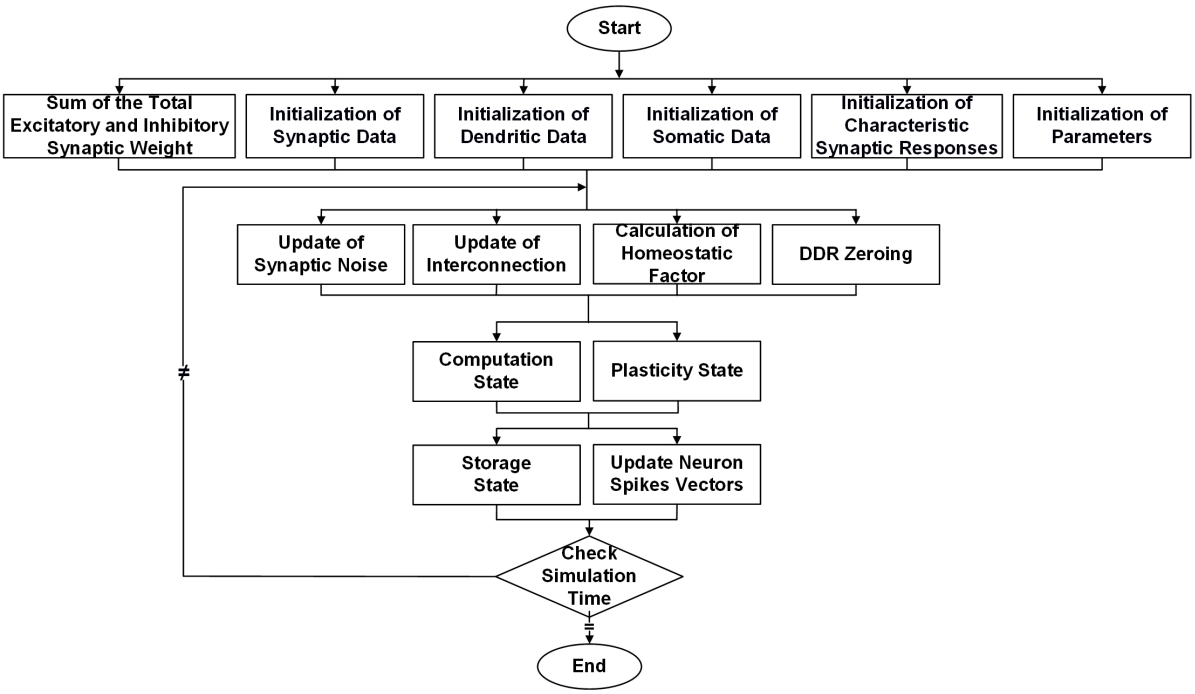

Fig. 4.9 Flow chart of the system functions

4.10 Architecture characteristics

In summary, the system is divided into four general phases based on the most time-consuming functions. At first, the initialization of the system consumes time, based on the size of the network. The total consumed time of the second phase, the synaptic update, depends on the percentage of the interconnections and the size of the network. In the third and fourth phase, the calculation of the neurons and the data storage are taking place, respectively. The system is ready to simulate a maximum number of 8192 groups of synapses per neuron. The total number of dendrites is up to 64 per neuron and the total number of neurons are up to 60.

The system has significant parallelism in the neuron computation because all the neurons calculate their state in parallel. An advantage of this design is that the system performance isn't effected from the scale of the neurons. The system is fully pipelined, so the update function requires (8192 + N) clock cycles to complete calculation of the neurons. The learning state is executed in parallel with the update state because of the pipelined architecture. Where N - relatively few - are the clock cycles required to calculate a synapse, including the change of weight.

On the other hand, the system is affected by the scale of the synapses and dendrites per neuron because the size of internal memory was designed to serve a specific number of dendrites per neuron and synapses per dendrite. The design decision, as in section 3.5 (the system dimensioning) referred, limits the system and the reading of the Block Ram is unfeasible even if all the synapses are inactive. The inactive neurons do not affect the system because of the parallelism in the neuron levels. Therefore, we have equipped the architecture with the maximum number of neurons that can be simulated in the specific FPGA.

4.11 Architecture integration and resource consumption

The architecture was implemented on the Convey HC-2ex hybrid FPGA super-computer, because of the large amount of data that should be stored in external memory and the required memory bandwidth. The Convey HC-2ex has four Virtex-6 LX760 FPGAs and all FPGAs share the same memory and one coprocessor. The platform has 14 memory controllers for parallel read and store. Our implementation is integrated into the four FPGAs of the platform.All the memory controllers are operated, when we read and store data from the memory, in order to take advantage of the memory bandwidth.

The development of the system was done, using the tool Xilinx ISE Design 12.4. In order to design the units we used hardware description language VHDL. The different functions and the overall design were simulated and validated with the ISE Modelism 12.4 (M.81d).

Table 4.1 Resource utilization and clock frequency

	Convey HC-2 Virtex-6 LX760 FPGA
Number of occupied slices	52%
Number of DSPs	28%
Number of BRAM/FIFO	99%
Clock Frequency	150MHz

Table 4.2 Resource utilization and clock frequency with curve fitting technique

	Convey HC-2 Virtex-6 LX760 FPGA
Number of occupied slices	80%
Number of DSPs	70%
Number of BRAM/FIFO	79%
Clock Frequency	150MHz

The connection of the design with platform Convey HC 2ex, was implemented in C language code, Assembly and Verilog. The system was fully developed, implemented, and evaluated experimentally. The table 4.1 shows the resource utilization of the design, after Synthesis and Place & Route, in a Virtex-6 LX760 FPGA of the hybrid platform as well as the clock frequency for the particular design. It should be clarified that a percentage of the overall consumption is caused by the platform Convey. The reason is that the platform uses logic and BRAMs to implement the interface of the Memory Controller. The percentage is increased when the system uses more memory controllers due to the increase of complexity. Our system uses 16 MCs , so the consumption is the maximum. The percentages confirm that the limiting element of the design was the number of BRAM.

Table 4.2 shows the resource utilization and the clock frequency of the design with the curve fitting technique. we didn't add additional neurons in this system because of additional changes to be made to operate the simulator correctly. However, the resources demonstrate that they can simulate additional neurons and extend the network. It is observed that the slices have significantly increased because the polynomial coefficients are stored in registers.

Chapter 5

Results

This chapter discusses the experimental setup and the performance in compare with similar simulations that were implemented in software. In addition, this chapter shows some experimental results of the biological simulations through a specific graphical representation.

5.1 Experimental results

The simulator, through the host, reads the input data for the initialization and the topology of the network. The output results are stored in the external memory when each simulation step is completed. At the end of the simulation, the experimental results (text files) are extracted and the user is informed. The Chapter 3 presents the complete application and the visualization of the results. The entire process of the simulator (interface and hardware) is automated and the results are presented in 2d/3d representations. Next, we will show some experimental results of the simulation with spike raster plots that are graphical representations of the neuron activity-spike timing. Raster plots are extensively used by biologists.

Figure 5.1 shows an experimental example of the SNN simulation executed on our system. The network consists of 120 neurons, 3038 dendrites and 202300 synapses and runs for 100ms simulation time. The configuration of the network is :

- 1,5ms refractory period
- 2ms synaptic activation delay
- distribution of excitatory synapses is 80% and inhibitory is 20%
- distribution of excitatory neurons 80% and inhibitory is 20%
- partially connected network (10%)

- background synaptic noise 100Hz
- BCM thresholds, E= 2mV and TH= 4mV
- BCM rates, LTP=2 LTD=3
- homeostatic rate 0.1

We also gave a stimuli in a subset of neurons with duration 0.1ms. The plot shows the fires of the neurons throughout the duration of the simulation. Thus, we distinguish that some neurons are less active, compared with other neurons. Also, the refractory period of the neuron and the synaptic activation delay are clearly visible.

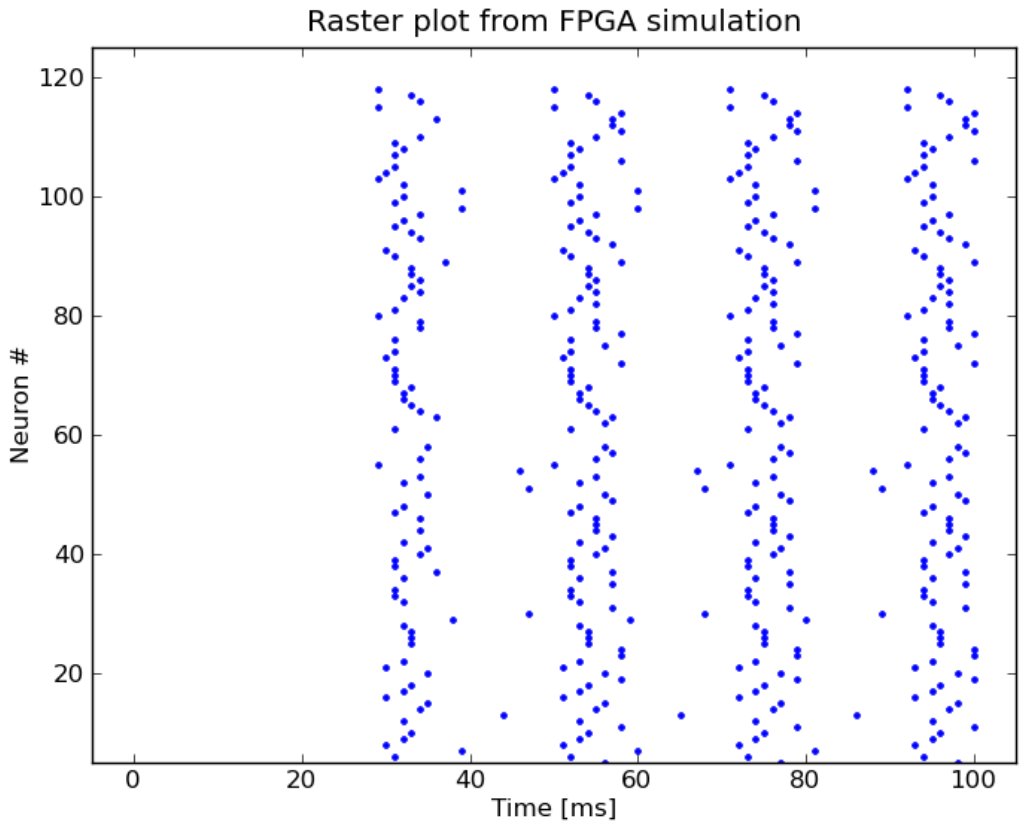

Fig. 5.1 Raster plot for 120 neurons and 100 ms actual simulation executed on the FPGA - based platform

The figures 5.3 5.2 below are some additional simulations, of duration 100ms, with different topology but the configuration was the same, as mentioned above.

In addition, we run a small network with 5 neurons and a few synapses. Then, we randomly select some synapses to indicate the potentiation and depression of the synaptic strength.

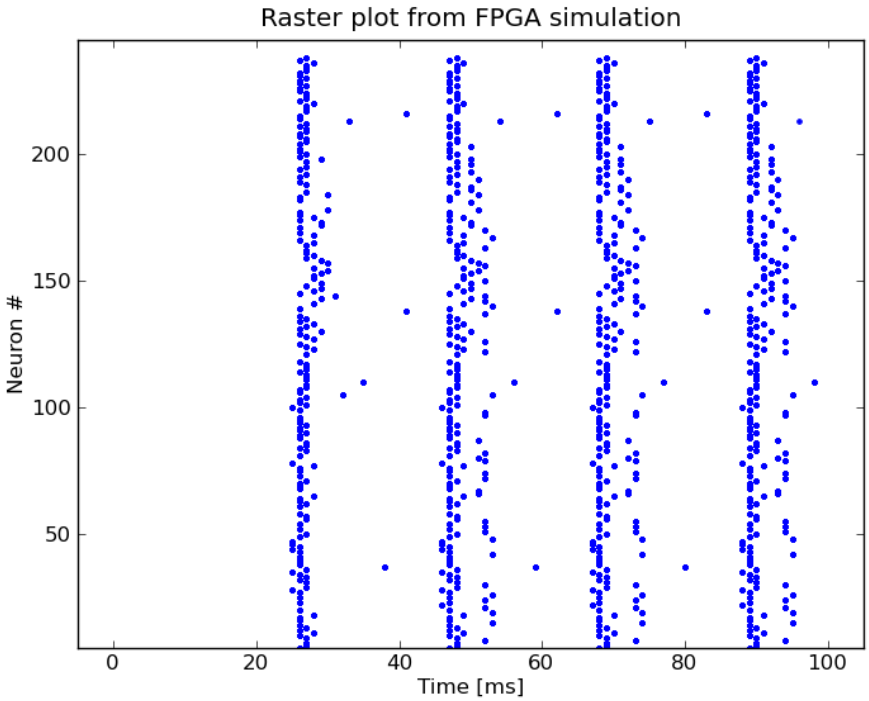

Fig. 5.2 Raster plot for 240 neurons and 100 ms actual simulation executed on the FPGA - based platform

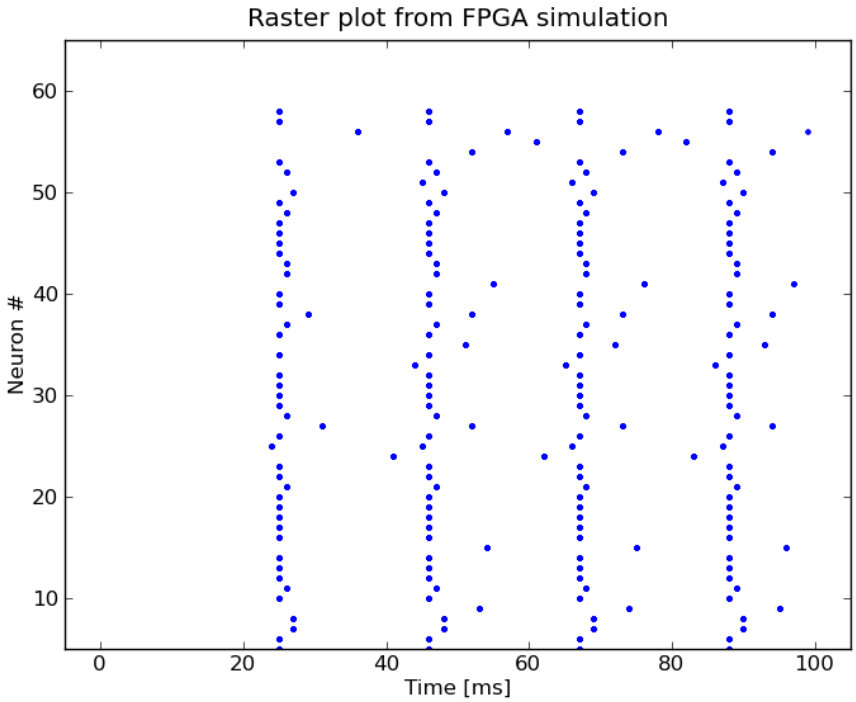

Fig. 5.3 Raster plot for 60 neurons and 100 ms actual simulation executed on the FPGA - based platform

Figures 5.4 5.6 5.7 show the changes of the synaptic weights during the plasticity rules. We made an experiment, only with BCM rule and another one for both the BCM and homeostatic rule together. We observed that the homeostatic plasticity influences the synaptic weights by limiting them at the lower levels.

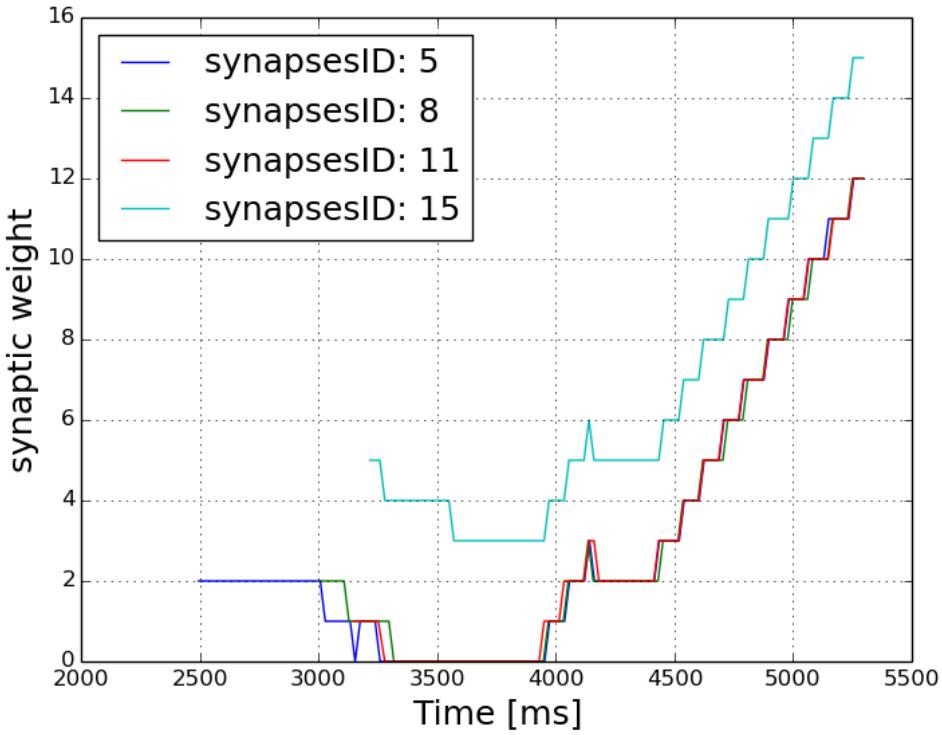

Fig. 5.4 Changes of the synaptic weights with BCM rule, neuronID=1, dendriteID=2

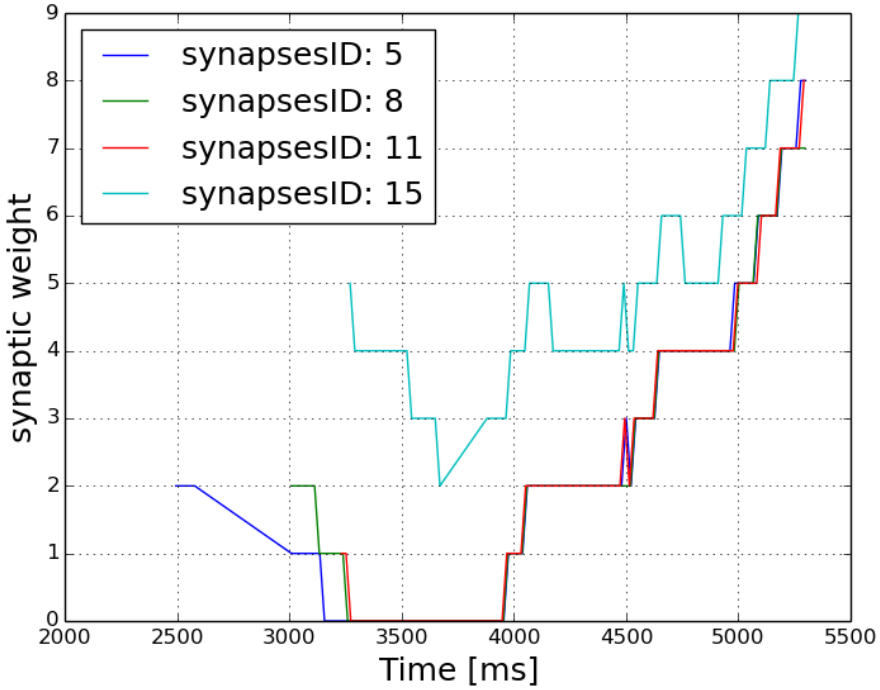

Fig. 5.5 Changes of the synaptic weights with BCM rule and homeostatic plasticity, neuronID=1, dendriteID=2

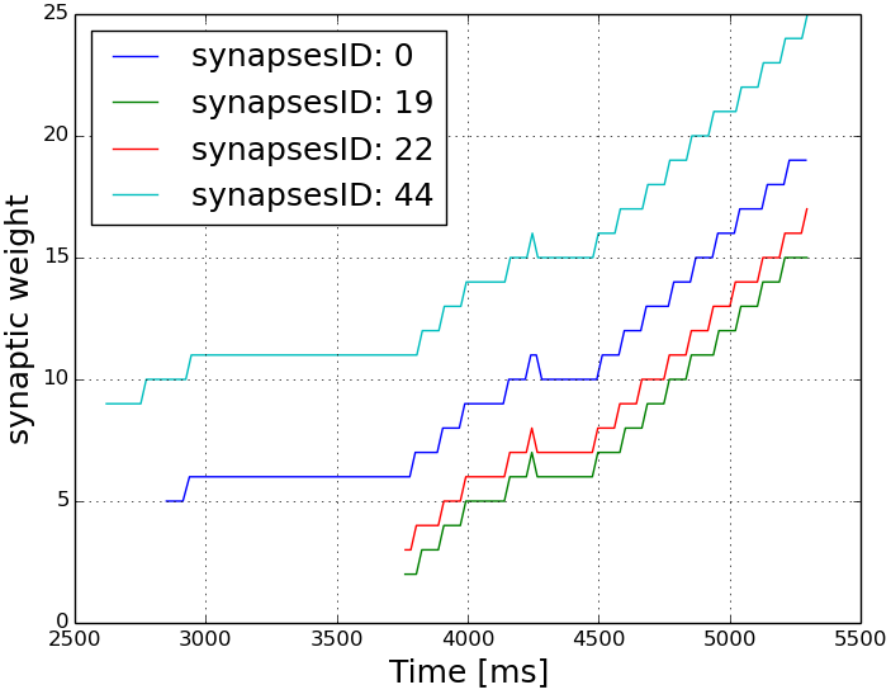

Fig. 5.6 Changes of the synaptic weights with BCM rule, neuronID=4, dendriteID=3

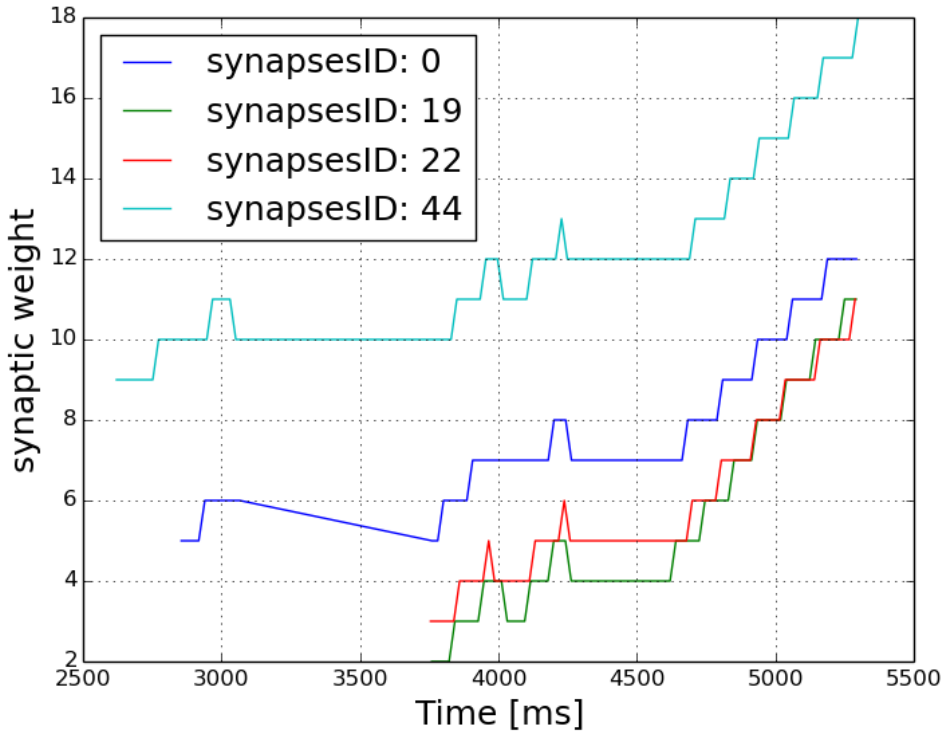

Fig. 5.7 Changes of the synaptic weights with BCM rule and homeostatic plasticity, neuronID=4, dendriteID=3

5.2 System validation and Performance Comparison

The researchers of the computational biology laboratory [36] have studied and validated the experimental results. The performance of the system which was impemented in hardware, was compared with similar tests implemented on software.

Table 5.1 shows the execution time of a network with 240 neurons, 12144 dendrites, 809825 synapses and 0.1ms simulation step in hardware vs. the software simulator BRAIN on an Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz, which is the state-of-the-art simulator. The result is that the FPGA implementation of the network on a multi-FPGA system with 4 Virtex-6 LX760 FPGA provides a speedup of 877-923X times of similar neuron networks simulations in software on a CPU with 6 cores at 2.10GHz.

It was not possible to compare our work to other neural network architectures implemented on FPGA technology due to the substantially more complex model in our work and the modeling of the BCM and homeostatic plasticity that have not been reported in another work, to date.

Table 5.1 Execution time 240 neurons and 0.1ms simulation time in FPGA vs. a CPU with 6 threads at 2.10GHz

Simulation time	Execution Time(HW)	Execution Time(SW)	Speedup
5sec	13,39 sec	12371 sec	923x
30sec	80,48 sec	71647 sec	890x
60sec	160,95 sec	141286 sec	877x

Chapter 6

Conclusions and future work

The last chapter discusses a final assessment of the design and presents recommendations for the future. Our aim is to improve the existing implementation, both in performance and in biological level.

6.1 Conclusions

This thesis describes and analyzes a network of biological neurons. The aim was to create a configurable simulator which will be integrated in the hybrid super-computer Convey HC-2ex and allows the remote execution of the model. This thesis describes how the exploitation of the advantages of reconfigurable logic, and especially the internal bandwidth of BRAMs, resulted in the design of a detailed neural network models for biological functions of the brain cells at the level of the synapses, and how sparse interconnections can be efficiently modeled in external memory. The Convey platform had an important role in the final result, due to the advanced interface for communicating FPGA to the external memory. The main contribution of this work was the detailed nature of the neural network model that we implemented, which is substantially more complex in comparison with other implementations.

6.2 Future Work

In the future we will further develop the model with ever-increasing degree of accuracy, as well as size the architecture for different types of neurons, depending on the brain functions that we want to simulate. Also, we could implement different spiking neural networks (e.g. HH model, Izhikevich) and additional learning mechanisms, so the biologists will have a complete application to run their different type of models.

A notable expansion of this work could be a different modeling of the inhibitory neurons. In the current architecture, each neuron has committed BRAMs that can host up to 64 dendrites. However, the inhibitory neurons have only one dendrite, so it is waste of resources to occupy the total of BRAMs. Hence, an architecture that can manage the inhibitory and excitatory neurons, could be designed.

The current architecture takes full advantage of the system. However, the implementation and comparison of the system in a different platform such as Maxeler computer could be a future work.

References

[1] Beuler, M., Tchaptchet, A., Bonath, W., Postnova, S., and Braun, H. A. (2012). Real-time simulations of synchronization in a conductance-based neuronal network with a digital fpga hardware-core. In *International Conference on Artificial Neural Networks*, pages 97–104. Springer.

[2] Bienenstock, E. L., Cooper, L. N., and Munro, P. W. (1981). Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. Technical report, DTIC Document.

[3] BrainscaleProject (2011). Brainscales - brain-inspired multiscale computation in neuromorphic hybrid systems. http://brainscales.kip.uni-heidelberg.de.

[4] Branco, T. and Häusser, M. (2011). Synaptic integration gradients in single cortical pyramidal cell dendrites. *Neuron*, 69(5):885–892.

[5] Cheung, J. (2010). Neurons, nerve tissues and the nervous system. http://www.biomedicalengineering.yolasite.com/neurons.php.

[6] Cheung, K., Schultz, S. R., and Leong, P. H. (2009). A parallel spiking neural network simulator. In *Field-Programmable Technology, 2009. FPT 2009. International Conference on*, pages 247–254. IEEE.

[7] Cheung, K., Schultz, S. R., and Luk, W. (2012). A large-scale spiking neural network accelerator for fpga systems. In *International Conference on Artificial Neural Networks*, pages 113–120. Springer.

[8] Cheung, K., Schultz, S. R., and Luk, W. (2015). Neuroflow: A general purpose spiking neural network simulation platform using customizable processors. *Frontiers in neuroscience*, 9.

[9] Crasto, C. J., Koslow, S. H., Bower, J. M., and Beeman, D. (2007). Constructing realistic neural simulations with genesis. *Neuroinformatics*, pages 103–125.

[10] Delorme, A. and Thorpe, S. J. (2003). Spikenet: an event-driven simulation package for modelling large networks of spiking neurons. *Network: Computation in Neural Systems*, 14(4):613–627.

[11] Deng, B., Zhang, M., Su, F., Wang, J., Wei, X., and Shan, B. (2014). The implementation of feedforward network on field programmable gate array. In *Biomedical Engineering and Informatics (BMEI), 2014 7th International Conference on*, pages 483–487. IEEE.

References

[12] Furber, S. B., Lester, D. R., Plana, L. A., Garside, J. D., Painkras, E., Temple, S., and Brown, A. D. (2013). Overview of the spinnaker system architecture. *IEEE Transactions on Computers*, 62(12):2454–2467.

[13] Gerstner, W. and Kistler, W. M. (2002). *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press.

[14] Goddard, N. H., Lynne, K. J., Mintz, T., and Bukys, L. (1987). *Rochester connectionist simulator*. Department of Computer Science, University of Rochester.

[15] Goodman, D. F. and Brette, R. (2009). The brian simulator. *Frontiers in neuroscience*, 3:26.

[16] Graas, E., Brown, E., and Lee, R. H. (2004). An fpga-based approach to high-speed simulation of conductance-based neuron models. *Neuroinformatics*, 2(4):417–435.

[17] Grassmann, C. and Anlauf, J. K. (1999). Fast digital simulation of spiking neural networks and neuromorphic integration with spikelab. *International Journal of Neural Systems*, 9(05):473–478.

[18] Harel, Z. S. (2007). Models of synaptic plasticity. http://www.scholarpedia.org/article/Models_of_synaptic_plasticity.

[19] Hartmann, G., Frank, G., Schäfer, M., and Wolff, C. (1997). Spike128k-an accelerator for dynamic simulation of large pulse-coded networks. In *Proceedings of MicroNeuro*, volume 97, pages 130–139.

[20] Hebb, D. O. (1949). The organization of behavior: A neuropsychological theory.

[21] Hellmich, H. H. and Klar, H. (2004). See: a concept for an fpga based emulation engine for spiking neurons with adaptive weights. In *5th WSEAS Int. Conf. Neural Networks Applications (NNA'04)*, pages 930–935.

[22] Hines, M. L. and Carnevale, N. T. (1997). The neuron simulation environment. *Neural computation*, 9(6):1179–1209.

[23] Hodgkin, A. L. and Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500.

[24] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558.

[25] Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572.

[26] Jahnke, A., Roth, U., and Klar, H. (1996). A simd/dataflow architecture for a neurocomputer for spike-processing neural networks (nespinn). In *Microelectronics for Neural Networks, 1996., Proceedings of Fifth International Conference on*, pages 232–237. IEEE.

[27] Kousanakis, E. (2015). Neural network simulation speedup based on reconfigurable logic. diploma thesis", Electrical and Computer Engineering.

[28] Leung, B., Pan, Y., Schroeder, C., Memik, S. O., Memik, G., and Hartmann, M. J. (2008). Towards an "early neural circuit simulator": A fpga implementation of processing in the rat whisker system. In *Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on*, pages 191–196. IEEE.

[29] Markram, H., Toledo-Rodriguez, M., Wang, Y., Gupta, A., Silberberg, G., and Wu, C. (2004). Interneurons of the neocortical inhibitory system. *Nature reviews. Neuroscience*, 5(10):793.

[30] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

[31] Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., Jackson, B. L., Imam, N., Guo, C., Nakamura, Y., et al. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673.

[32] Moore, S. W., Fox, P. J., Marsh, S. J., Markettos, A. T., and Mujumdar, A. (2012). Bluehive-a field-programable custom computing machine for extreme-scale real-time neural network simulation. In *Field-Programmable Custom Computing Machines (FCCM), 2012 IEEE 20th Annual International Symposium on*, pages 133–140. IEEE.

[33] Mytakis, D. (2016). Real-time visualization of neuroscience model simulation. diploma thesis", Electrical and Computer Engineering.

[34] Oja, E. (1982). Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273.

[35] Pearson, M. J., Pipe, A. G., Mitchinson, B., Gurney, K., Melhuish, C., Gilhespy, I., and Nibouche, M. (2007). Implementing spiking neural networks for real-time signal-processing and control applications: A model-validated fpga approach. *IEEE Transactions on Neural Networks*, 18(5):1472–1487.

[36] PoiraziLab (2014). http://www.dendrites.gr.

[37] Roggen, D., Hofmann, S., Thoma, Y., and Floreano, D. (2003). Hardware spiking neural network with run-time reconfigurable connectivity in an autonomous robot. In *Evolvable hardware, 2003. proceedings. nasa/dod conference on*, pages 189–198. IEEE.

[38] Ros, E., Ortigosa, E. M., Agís, R., Carrillo, R., and Arnold, M. (2006). Real-time computing platform for spiking neurons (rt-spike). *IEEE Trans. Neural Networks*, 17(4):1050–1063.

[39] Schoenauer, T., Mehrtash, N., Jahnke, A., and Klar, H. (1999). Maspinn: Novel concepts for a neuroaccelerator for spiking neural networks. In *Ninth Workshop on Virtual Intelligence/Dynamic Neural Networks: Neural Networks Fuzzy Systems, Evolutionary Systems and Virtual Re*, pages 87–96. International Society for Optics and Photonics.

[40] Sejnowski, T. (1977). Statistical constraints on synaptic plasticity. *Journal of theoretical biology*, 69(2):385–389.

[41] Smaragdos, G., Isaza, S., van Eijk, M. F., Sourdis, I., and Strydis, C. (2014). Fpga-based biophysically-meaningful modeling of olivocerebellar neurons. In *Proceedings of the 2014 ACM/SIGDA international symposium on Field-programmable gate arrays*, pages 89–98. ACM.

[42] Spruston, N. (2008). Pyramidal neurons: dendritic structure and synaptic integration. *Nature Reviews Neuroscience*, 9(3):206–221.

[43] Stent, G. S. (1973). A physiological mechanism for hebb's postulate of learning. *Proceedings of the National Academy of Sciences*, 70(4):997–1001.

[44] Thomas, D. and Luk, W. (2009). Fpga accelerated simulation of biologically plausible spiking neural networks. In *Field Programmable Custom Computing Machines, 2009. FCCM'09. 17th IEEE Symposium on*, pages 45–52. IEEE.

[45] Upegui, A., Peña-Reyes, C. A., and Sanchez, E. (2003). A functional spiking neuron hardware oriented model. In *International Work-Conference on Artificial Neural Networks*, pages 136–143. Springer.

[46] Wang, R. M., Cohen, G., Stiefel, K. M., Hamilton, T. J., Tapson, J. C., and van Schaik, A. (2013). An fpga implementation of a polychronous spiking neural network with delay adaptation. *Frontiers in neuroscience*, 7:14.

[47] Wolff, C., Hartmann, G., and Ruckert, U. (1999). Parspike-a parallel dsp-accelerator for dynamic simulation of large spiking neural networks. In *Microelectronics for Neural, Fuzzy and Bio-Inspired Systems, 1999. MicroNeuro'99. Proceedings of the Seventh International Conference on*, pages 324–331. IEEE.

[48] y Cajal, S. R. (1909). Histologie du système nerveux de l'homme et des vertébrés, édition française revue et mise à jour par l'auteur, traduite de l'espagnol par l. azoulay.

[49] Zell, A., Mache, N., Huebner, R., Mamier, G., Vogt, M., Schmalzl, M., and Herrmann, K.-U. (1994). Snns (stuttgart neural network simulator). In *Neural Network Simulation Environments*, pages 165–186. Springer.

[50] Zhu, J. and Sutton, P. (2003). Fpga implementations of neural networks–a survey of a decade of progress. In *International Conference on Field Programmable Logic and Applications*, pages 1062–1066. Springer.