Implementation of neural plasticity mechanisms on reconfigurable hardware for robot learning

Ilias Alevizos

Master of Philosophy

Institute of Perception, Action and Behaviour
School of Informatics
University of Edinburgh

2011

Abstract

It is often assumed that insects are "primitive" animals, without the ability to exhibit complex learning behaviour. Fortunately, their tiny brains quite often surprise us with their performance. This thesis investigates the plasticity mechanisms of the insect brain through the research method of neurorobotics, i.e., the development of a physical agent, equipped with a silicon brain.

In order to implement such a brain, we have chosen to model it directly onto hardware. Not only does this allow us to take advantage of the inherent hardware parallelism, but the robot can also behave in a completely autonomous mode, without having to communicate with the software simulator of a remote machine. FPGAs offer both the option for such a low-level design approach and the flexibility required in computational studies of biological neural networks. With the use of VHDL (a hardware description language), we develop a simulator for neural networks, designed as a series of computational modules, running in parallel and solving the differential equations which describe neural processes. It has the ability to simulate networks with spiking neurons that follow a phenomenological model, proposed by Izhikevich, which requires only 13 operations per 1 ms of simulation. The synaptic plasticity mechanism can be either that of spike timing-dependent plasticity (STDP) or a modified version of STDP which is also affected by neuromodulators. There are no constraints, as far as the connectivity pattern is concerned. The hardware simulator is then added as a peripheral to an embedded system so that it can be more easily controlled through software and connected to a robot. We show that this hardware system is able to model networks with hundreds of neurons and with a speed performance that is better than real-time. With some slight modifications, it could also scale up to thousands of neurons, starting to approach the size of the insect brain.

Subsequently, we use the simulator in order to model a neural network with an architecture inspired by the insect brain, representing the connectivity of the antennal lobe, the mushroom body and the lateral horn, structures which are part of the insect's olfactory pathway. Our silicon brain is then attached to a robot and its limits and capabilities are tested in a series of experiments. The experiments involve tasks of associative learning inside an arena which is based on a T-maze set-up usually employed in behavioural experiments with flies. The robot is trained to associate different stimuli (or combinations of stimuli) with a punishment, as indicated by the presence of a light source. We observe that the robot can solve most of the tasks, including elemental learning, discrimination learning, biconditional discrimination and negative patterning but fails to solve the problem of positive patterning. It is concluded that the architecture of the insect's olfactory pathway has the computational efficiency to solve even non-elemental learning tasks. However, this pattern of results does not precisely match the fly, suggesting we have not fully understood the learning mechanisms involved. Moreover, embedding the learning circuit in robot behaviour reveals that the simple version of STDP is

not the appropriate mechanism which can link neural plasticity to learning behaviour. Although the modified version of STDP is more suitable, it remains problematic as well as sensitive to timing issues. Therefore, we propose that STDP might function more as a “priming” process rather than as the basic learning mechanism.

Acknowledgements

Usually, acknowledgements are considered to be a gesture of gratitude towards those being acknowledged. Or maybe, they are more like charity. The more one is willing to grant them, the more fervently he ascertains ownership over “his” work/property. However, I will indulge myself in the luxury of thanking two people. My supervisor, Barbara Webb, for giving me the freedom to follow my own path while, at the same time, helping me to make my ideas more concrete. And Jan Wessnitzer, for all the helpful discussions.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Ilias Alevizos*)

Table of Contents

List of Figures

List of Tables

Nomenclature

AC	Antennal Commissure
ACT	Antennocerebral Tract
AL	Antennal Lobe
AN	Antennal Nerve
APL	Anterior Paired Lateral
ARM	Anaesthesia-resistant Memory
ASIC	Application Specific Integrated Circuit
cAMP	Cyclic Adenosine Monophosphate
CR	Conditioned Response
CS	Conditioned Stimulus
DPM	Dorsal Paired Medial
EN	Extrinsic Neuron
EPSP	Excitatory Post Synaptic Potential
FIFO	First In, First Out
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GABA	gamma-Aminobutyric Acid
GPU	Graphics Processor Unit
IR	Infra Red
KC	Kenyon Cell

LH	Lateral Horn
LHCN	Lateral Horn Comparison Neuron
LHI	Lateral Horn Interneuron
LI	Local Interneuron
LTM	Long-term Memory
LUT	Look-up Table
MB	Mushroom Body
MIF	Memory Initialization File
MTM	Medium-term Memory
OR	Odorant Receptor
ORN	Olfactory Receptor Neuron
OSN	Olfactory Sensor Neuron
PN	Projection Neuron
RAM	Random Access Memory
STDP	Spike Timing Dependent Plasticity
STM	Short-term Memory
SVM	Support Vector Machine
UR	Unconditioned Response
US	Unonditioned Stimulus
VHDL	Very-high-speed integrated circuits Hardware Description Language
VLSI	Very Large Scale Integration
VN	Value Neuron
VNC	Ventral Nerve Cord

Noon
The tree gathers
the shadow upon its branches

P. Kyparissis

Chapter 1

Introduction

Explaining or, more precisely, interpreting adaptive behaviour of animals and endowing animats with such an adaptive behaviour have been long-sought goals of the fields of neuroscience and robotics respectively. It is self-evident that approaching either one of these goals would have a significant impact on the respective field. Roboticists have developed the field of bioinspired robotics by closely inspecting the mechanisms that biological organisms employ in order to solve specific problems and then "copying" them onto robots. However, biologists have been reluctant to move towards a roboinspired biology, although a collaboration between the two research areas could result in a positive feedback loop from which both could benefit (Webb, 2001). The work presented in this thesis lies at this intersection of robotics and neuroscience, trying to address both the issue of animal learning and that of building robots which can learn from their experience in real world situations.

It has been argued that learning and skilful action are the distinctive features of intelligent behaviour (Dreyfus, 2002). It is not surprising therefore that a substantial body of research has been devoted to discovering exactly how animals, including humans, acquire new skills. Although experiments with mammals, such as primates and rats, can provide useful insights, the tools that are available today do not seem suitable enough for a detailed investigation of the neural mechanisms of learning in such animals. Instead, this thesis is going to use an insect, namely *Drosophila melanogaster*, as the animal model and hypothesis testing platform since it can prove advantageous from several points of view. First, the size of the Drosophila brain is several orders of magnitude smaller than that of mammals, rendering it much more "tractable". Moreover, there exist genetic tools with which an impressive degree of control of even individual neurons is possible. On the other hand, the simplicity and accessibility of insect brains do not prevent them from exhibiting learning capabilities beyond those of mere reflexive responses. Equally important, at least for the purposes of this thesis, is the fact that such a brain does not impose insurmountable obstacles, as far as its modelling on hardware is concerned.

This is one point which distinguishes our work from previous studies in the field of neurorobotics. Most studies which include a brain model in their robotic platforms, especially those that aim for a biologically plausible model, develop their simulators in software and establish a communication link between the robot and the machine which runs the simulation. On the contrary, the aim of the present thesis is to develop a hardware simulator by taking advantage of the relatively recent technology of FPGAs. Our robot could thus function in a completely autonomous manner. Notwithstanding the hardships one must endure when working directly with hardware and proprietary technology, we hope to show that FPGAs have the potential to become a very useful tool for neuroroboticists, once a simple user interface has been built.

Moreover, our simulator should exhibit a speed performance that is close or better than real-time so that it may be used in robotic experiments. Using this hardware simulator, a simplified model of the fly's olfactory pathway is implemented and attached to a robot. The aim is to investigate how the model behaves when it has to learn associations in a closed-loop environment. By comparing the performance of the robot with that of flies in behavioural experiments, we attempt to conceive and propose ways with which our current theories and understanding of the fly's neural plasticity mechanisms could be amended in order to account for the observed discrepancies. The discussion of these issues will hopefully show that the methodology of neurorobotics may offer insights and new ways of perceiving and approaching the problems of computational neuroscience.

The thesis begins with a chapter (chap. 2) that reviews the research conducted until now on insect learning and memory, focusing mostly on flies. The relevant modelling studies are also reviewed, both software and hardware. Chapter 3 presents the digital design of the hardware simulator and the embedded system which supports it. Chapter 4 describes the robotic platform and discusses the results from the experiments with the robot. We conclude with chapter 5 which gives an overview of the possible future research paths, based on the present work.

Chapter 2

Literature review

The presentation of previous work on insect (mostly fly) learning and memory in this chapter follows a top-down approach. Beginning with more abstract and psychological theories of learning, we proceed with the neurobiology of the insect brain and behavioural studies of learning. Finally, the relevant modelling studies, both software and hardware, are discussed.

2.1 The psychology of animal learning

The literature on theories of learning and memory is vast and keeps expanding as new tools become available. It is not possible to cover it here. In fact, it is very hard even to simply mention all the relevant theories, studies and discussions. Instead, we restrict ourselves to associative theories of animal learning since these are usually tested in behavioural experiments with insects. For a more complete review, see (Pearce and Bouton, 2001) and (Schmajuk, 2008).

The Rescorla-Wagner theory (Rescorla and Wagner, 1972) is the dominant theory that tries to explain associative learning in psychological terms and is summarized in the next equation :

$$\Delta V_A = \alpha\beta(\lambda - V_T) \tag{2.1}$$

According to this theory, the change in the associative strength of a stimulus on any trial (ΔV_A) is proportional to the discrepancy between an asymptotic value of the magnitude of the US (λ) and the sum of the associative strengths of all the stimuli present on the trial (V_T). The parameters α and β are learning rate parameters determined by the salience of the CS and the characteristics of the reinforcer respectively. It is a theory that can explain many experimentally observed phenomena, like blocking [1], whereas its several modifications and expansions can provide for even more explanatory power.

[1] In which learning about a stimulus A first and then about the compound AB results in a weakened response to B alone.

For example, in order to explain latent inhibition [2], Mackintosh proposed a role for attention (Mackintosh, 1975). If $(\lambda - V_A) > (\lambda - V_X)$, where V_A is the associative strength of stimulus A and V_X the sum of the associative strengths of all stimuli except A, then it is assumed that A is a good predictor of the US and more attention is paid to it. Later, Pearce and Hall made the distinction between different forms of attention (Pearce and Hall, 1980). During learning, attention paid to the CS serves to strengthen its associability and make it a good predictor of the US. After learning, attention is still paid to the CS but only in order to produce the response. The fact that it has become a good predictor of the US actually decreases the attention paid to it. In order to address the issue of how stimuli are internally represented, Wagner proposed the model of standard operating procedures (SOP) (Wagner, 1981). Stimuli can be in a state of low, high or no activation and the established associations are determined by the state of the involved stimuli.

The assumption behind these theories is that we can predict the response of a compound stimulus simply by adding the responses to each of the constituent elements, thus the name elemental theories of learning. Due to the difficulties of elemental theories to explain several phenomena where combinations of stimuli are involved, the so called configural theories have been proposed. As their name suggests, they assume that, besides representations of single elements, there exist representations of combinations of elements, acting independently of their constituents. As we'll see later in chapter 4, the model of the insect brain for our robot implements a version of the configural cue theory. An important difference though is that the theories described above can also learn (or more precisely "unlearn") in every trial, even in the absence of the US ($\lambda = 0$). On the contrary, our neural network modifies its synapses only in those training trials in which the US is present.

2.2 Insect brain anatomy and neurophysiology

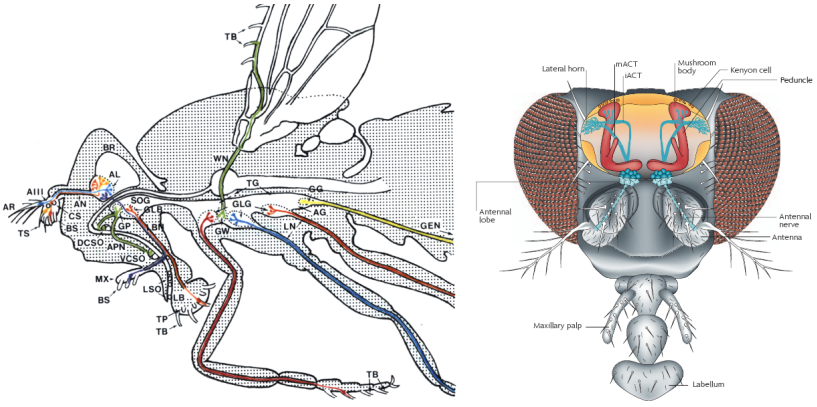
Before presenting any behavioural studies and the neural correlates of insect learning, we discuss first the necessary, neurobiological framework. Due to the fact that the modality of olfaction is usually targeted in these studies, the following subsections focus on the olfactory pathway. Although we refer generally to the insect brain, it should be noted that our model organism is the fly brain and that differences between the various insect species do exist. It is not yet known how exactly they might affect behaviour and for this reason, we will insist on referring to a typical insect brain.

[2] Learning of the CS (generation of the CR) is inhibited when the CS-US trials have been preceded by preexposure to the CS alone.

2.2.1 General view and anatomy

The insect brain is composed of the supraesophageal ganglion and the subesophageal ganglion. The rest of the insect nervous system, called the ventral nerve cord (VNC), runs through its body, from the thorax to the abdomen. A distinctive feature of it, compared to mammals, is the formation of ganglia (the segmental ganglia) at several points of its route which can function in a more or less autonomous way. The segmental ganglia are connected with bundles of axons.

The subesophageal ganglion of the brain (also called the posterior brain) provides a link between the supraesophageal ganglion and the segmental ganglia whereas the supraesophageal ganglion (also called the anterior brain) constitutes by far the largest part of the brain. Three main regions can be distinguished in the supraesophageal ganglion. The protocerebrum is the most prominent and hosts those structures, such as the mushroom bodies, which are usually related to information processing and control of higher functions. The optic lobes also belong to the protocerebrum. The deutocerebrum consists of the antennal lobe and the dorsal lobe (or antennal mechanosensory and motor centre). The antennal lobe receives input from the olfactory receptor neurons in the third antennal segment and the dorsal lobe from mechanosensory neurons in the basal antennal segments (Homberg et al., 1989). The smallest region of the supraesophageal ganglion is the tritocerebrum whose role lies mainly in taste perception and control of mouthparts (Rajashekhar and Singh, 1994).



(a) Various structures of the insect nervous system (Stocker, 1994)

(b) The main components of the olfactory pathway (Keene and Waddell, 2007)

Figure 2.1: The insect brain

Since the olfactory pathway has been studied more extensively, both in neurophysiological and in behavioural experiments, at least as far as learning and memory are concerned, we will mostly focus on it in the remainder of this section. This pathway starts in the antennae and the maxillary palps (fig. 2.1) where the olfactory receptor neurons are located. Their axons

bundle together to form the antennal nerve (AN) and project to the antennal lobes. There, they synapse with the dendrites of projection neurons (PN) which in turn organize themselves into antennocerebral tracts (ACT) and terminate in the calyces of the mushroom bodies (MB) and the lateral horns (LH) (fig. 2.2). The neurons of the mushroom bodies are called Kenyon cells (KCs) and have dendrites in the calyces, receiving input from PNs. Their axons extend to create the penduculus which splits to form several lobes just dorsal to the ALs. The olfactory pathway and the functional role of the structures involved are discussed in more detail in the next subsections. Several reviews are available (Keene and Waddell, 2007), (Davis, 2005), (Margulies et al., 2005), (Davis, 2004), (Hallem and Carlson, 2004), (Heisenberg, 2003), (Heisenberg, 1998). The details presented in the next sections and the numbers given refer to the nervous system of *Drosophila*.

2.2.2 Olfactory receptor neurons

The transformation of odour stimuli to electrochemical signals takes place in the third antennal segment and the maxillary palps. The neurons responsible for this transformation are called olfactory receptor neurons (ORNs, or olfactory sensory neurons). In *Drosophila melanogaster*, around 1200 of them can be found in the antennae and another 120 in the maxillary palps (Shanbhag et al., 1999), (Shanbhag et al., 2000). ORNs are packed together in numbers of up to four within hair-like protrusions called sensilla.

The dendrites of ORNs host odorant receptors (ORs, seven-transmembrane-domain proteins) onto which odour molecules can bind. This activates G-protein-coupled second-messenger systems which give rise to electrical signals (Hildebrand and Shepherd, 1997), (Dobritsa et al., 2003). It is estimated that Drosophila has around 60 odorant receptor genes (Or genes). However, some of them do not express either in the maxillary palps or the antennae. Each ORN probably expresses only one of these genes or a very limited subset of them and each odorant receptor is expressed in 2 to 50 ORNs with the exception of the gene Or83b which is expressed in every ORN (Jones et al., 2005). Odorant receptors can respond to more than one odours but they can be identified by their response spectra which are unique (Hallem et al., 2004).

2.2.3 Antennal lobes and projection neurons

After the initial phase of olfactory perception, the next stage of the processing hierarchy is located at the antennal lobes (ALs). Signals are relayed to the ALs via the antennal nerve (AN), a collection of about 1700 axons, mainly from ORNs of the third antennal segment. Some of them (around 500) originate from mechanosensory receptors and do not terminate at the AL (Stocker et al., 1990). ALs constitute the first stage of bilateral integration since around 1000 axons of the AN cross over to the contralateral lobe through the antennal commissure (AC).

The ALs are organized into 43 subunits of almost spherical shape called glomeruli (Laissue et al., 1999), (Vosshall et al., 2000). It has to be noted that glomeruli do not host any cell bodies themselves but are instead sites where synapses are formed (Stocker et al., 1990). Cell bodies are located near the lobe but not within the glomeruli. Each of the afferent fibres of the AN targets only one specific type of glomerulus. Moreover, ORNs which express the same receptor project to a single specific glomerulus (Vosshall et al., 2000), (Gao et al., 2000). Two types of neurons comprise the postsynaptic targets of the AN axons, local interneurons (LIs) and projections neurons (PNs). LIs are axonless neurons, usually ramifying extensively throughout the entire lobe and can be either excitatory or inhibitory (Wilson and Laurent, 2005), (Shang et al., 2007). On the other hand, PNs mediate the transmission of information from the ALs to higher brain centres, such as the mushroom bodies (MBs) and the lateral horn (LH).

About 200 PNs, each having dendrites within only a single glomerulus, form with their axons the antennocerebral tracts (ACTs) (Wong et al., 2002), (Stocker et al., 1990). The numbers of the neurons indicate a considerable convergence of about 10:1 from the ORNs to the PNs. The inner ACT (iACT) connects the ALs with both the LHs and the calyces of the MBs (see subsection 2.2.4) whereas the middle and outer ACTs (mACT, oACT) project directly and almost exclusively to the LHs. Individual PNs innervate a substantial part of the LH but a certain stereotopy is evident. PNs from the same glomerulus have similar projection patterns and PNs with similar projection patterns prefer neighbouring glomeruli (Marin et al., 2002).

2.2.4 Mushroom bodies

Three main regions comprise the mushroom bodies, the calyx, the penduculus and the lobes (fig. 2.2). The MB neurons are divided in two classes, the intrinsic neurons which arborize solely within the MBs and the extrinsic neurons which link MBs with other brain areas. The most abundant intrinsic neurons which give MBs their characteristic shape (hence their name) are the Kenyon cells (around 2500 in Drosophila, a 1:10 divergence from PNs to KCs). They can be classified as γ, α'/β' or α/β neurons according to their projection patterns in the lobes. Their cell bodies lie in the posterior dorsal cortex of the brain (Tanaka et al., 2008). They extend their dendrites to the anterior to form the calyx which is thought to be a main input zone. This input is provided by the antennal lobe projection neurons, mainly through the inner antennocerebral tract (iACT) but also, to a much lesser extent, through the outer ACT (oACT) and the inner middle ACT (imACT) (Tanaka et al., 2008). Although in Drosophila the calyces receive input only from the olfactory pathway, in other species, such as honeybees, they can have visual or gustatory input as well (Fahrbach, 2006). It is also important to note that the calyces are not the only input area of the MBs since there are extrinsic neurons that innervate both the penduculus and the lobes (Tanaka et al., 2008), (Fahrbach, 2006). Therefore, it seems very probable that MBs are a site of multimodal convergence.

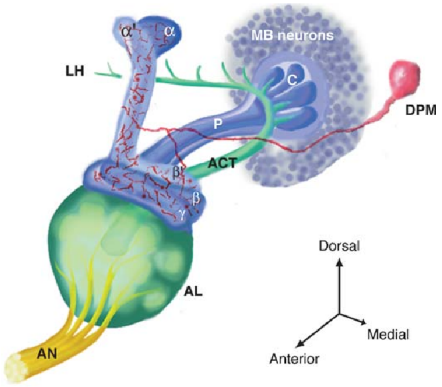


Figure 2.2: Inside the insect brain. The olfactory nervous system. (Davis, 2005)

The axons of the Kenyon cells bundle together in parallel fibres and run down the penduculus until they reach a bifurcation point where the lobes begin (fig. 2.2). The γ neurons are to be found at the periphery of the penduculus, the α/β neurons at the core and the α'/β' neurons in between. The penduculus is innervated both by intrinsic neurons (dorsal paired medial, DPM and anterior paired lateral, APL) and by several types of extrinsic neurons (Tanaka et al., 2008). The pair of DPM neurons, innervating the anterior part of the penduculus, have been shown to be important for memory stabilization (Waddell et al., 2000), (Keene and Waddell, 2007). As far as the others are concerned, they have been only recently discovered and their role remains unclear. In honeybees (but not in Drosophila), recurrent, GABAergic connections have been observed that link the lobes with both the calyx and the penduculus (Fahrbach, 2006), (Grunewald, 1999).

Research on the response properties of the KCs has not been extensive but the preliminary results available are interesting enough to deserve to be mentioned. When the olfactory pathway is not driven by any stimulus, the KCs remain remarkably quiescent (Perez-Orive et al., 2002). This is not a trivial fact, considering that, even in the absence of stimuli, projection neurons provide the calyx with a constant spontaneous input. When an odour is presented, the responses of KCs are both rare and sparse, thus rendering them highly informative (Turner et al., 2008), (Perez-Orive et al., 2002) (fig. 2.3). The response properties depend on the lobes of the KCs with the α'/β' lobes being the most responsive and broadly tuned (Turner et al., 2008). Besides the excitatory input from projection neurons, KCs are also inhibited by GABAergic lateral horn interneurons (LHIs) which arborize more diffusely and less selectively than the PNs in the calyx. The inhibitory drive from LHIs is almost out of phase relative to the excitatory drive from PNs and when pharmacologically blocked, results in a broadening of KCs tuning. These properties of KCs indicate that they may act as coincidence detectors, sparsening the odour space so that discrimination becomes more efficient (Turner et al., 2008),

(Laurent, 2002). Ca^{2+} imaging has confirmed the sparseness of the response patterns of MB neurons. It has also showed that MBs recruit different subsets of the KCs for different odours and that these subsets are conserved across individuals (Wang et al., 2004).

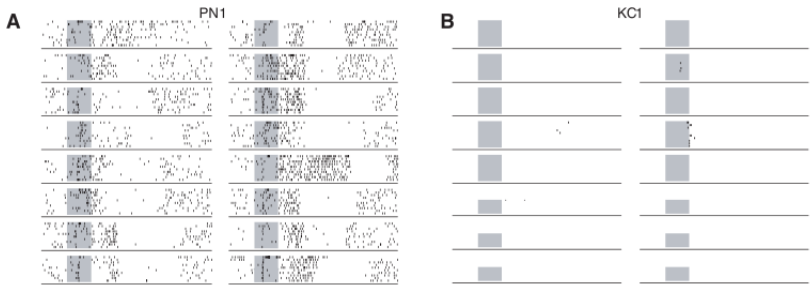


Figure 2.3: PNs' and KCs' responses to 16 different odours (Perez-Orive et al., 2002)

2.2.5 Key points

From the structures described above, we will later focus mostly on the MB. As explained in the next section (2.3), is seems to be the most crucial structure for learning and memory. We will thus attempt to incorporate the functionality of the MB in our model. One important feature of the olfactory pathway is the observed divergence of the neural signal as it passes from the PNs to the KCs. This allows the MB to separate stimuli more efficiently (see also the results from modelling studies in section 2.4). Moreover, the sparse and rare activity of the KCs, with the help of the LHIs, means that their responses are highly informative. For these reasons, our neural network will try to model this cooperation between the PNs, the LHIs and the KCs.

2.3 Behavioural studies

It is obvious that a way to measure performance is required, in order to study learning and memory in animals. The simplicity of the classical and instrumental conditioning paradigms makes them quite attractive and they have come to dominate the field of learning and memory in invertebrates. The experimental setup proposed by Tully and Quinn is considered a classic and meets with widespread use (Tully and Quinn, 1985). First, the flies are driven into a tube surrounded by an electrifiable grid. Odour currents can also be directed into this tube. Therefore, delivery of odours can be combined with electric shocks so that flies can make the association between the conditioned stimulus of an odour and the unconditioned of the shock. They can later be tested in a similar setup with two tubes, each "emitting" a different odour (with one of them obviously being the conditioned odour). Great emphasis has also been put on olfactory learning rather than visual or tactile. This is not surprising, considering the biological

importance that odours have for insects. Moreover, or maybe because of this same reason, the olfactory neural pathway has received much more detailed investigation than other sensory pathways. It is therefore easier to track the neural correlates of olfactory memory. Finally, research has mostly concentrated on a specific brain structure, the mushroom bodies (MB) (see (Strausfeld et al., 1998) for the reasons).

The picture that is consequently drawn is that of a field dominated by olfactory classical/instrumental conditioning paradigms and learning mediated by the MBs. In the next sections the emphasis will be on these aspects of insect learning behaviour. Some caution should nevertheless be exercised. Insects have capabilities for other complex forms of learning as well (e.g. visual learning (Liu et al., 1999), (Schubert et al., 2002), (Tang and Guo, 2001)) and the relation between the different sensory modalities and the MBs is not at all clear. For example, flies can learn a task with visual cues and the MBs seem to be involved (Tang and Guo, 2001). However, their MBs do not receive any projections from the optic lobes in contrast to honeybees which possess such projections and exhibit visual learning as well. Overgeneralizations should therefore be met with some caution.

2.3.1 The US pathway

One of the central questions is how exactly the unconditioned stimulus is mediated in the brain of an insect. Quite interestingly, it was found that a single neuron is sufficient to mediate reinforcement in experiments of appetitive conditioning (Hammer, 1993). The VUMmx1 neuron, which innervates the ALs, the lateral protocerebrum and the MB calyces, responds strongly when honeybees are provided with sucrose with a response that lasts longer than the US. Furthermore, substituting the presentation of reward with a direct depolarization of VUMmx1 could lead to learning. After training, the conditioned odour can trigger a response from VUMmx1 which implies that its responsiveness depends on previously established associations. It is believed that VUMmx1 neurons release octopamine since this neuromodulator, when injected in ALs or MB calyces (but not lateral protocerebrum), can essentially substitute for the US and induce learning (Hammer and Menzel, 1998).

As far as aversive learning is concerned, dopamine seems to be the neuromodulator of interest. In fact, one study revealed that octopamine is required only for appetitive learning whereas dopamine only for aversive (Schwaerzel et al., 2003). Through the use of imaging tools, it was revealed that dopamine neurons projecting to the MBs (and not those projecting to PNs) are activated when an electric shock is delivered to the animal. As is the case with VUMmx1, dopamine neurons seem to predict punishment by exhibiting an increased response to a conditioned odour (Riemensperger et al., 2005). An innovative technique, which allows for a light-induced activation of either dopaminergic or octopaminergic neurons, confirmed the above results by replacing the US with an artificial neuromodulator release (Schroll et al.,

2006).

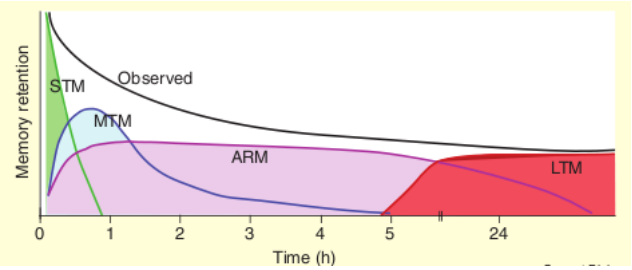
2.3.2 Memory phases

The so called "genetic dissection" of memory in Drosophila has led to a distinction between the various stages of memory formation (Tully et al., 1994) (for a review, see (Margulies et al., 2005)). Initially, there is an acquisition phase during the training procedure affected by genes such as *linotte*. Short-term memory (STM) appears during acquisition and has a lifespan of only a few minutes, disappearing completely within the first two hours after training. Mutations of the *dunce* and *rutabaga* (but not *amnesiac*) genes disrupts STM. These genes are highly expressed in the MBs and are involved in the cAMP signalling pathway. Restoring expression of *rutabaga* in the MBs of memory deficient *rut* mutants has been shown to be sufficient for rescuing STM (McGuire et al., 2003). Additionally, γ lobes seem to be more important in rescuing STM via *rutabaga* restoration (Zars et al., 2000). An STM trace also appears in the glomeruli of the ALs. Conditioning of an odour recruits more PN synapses, resulting in the activation of additional glomeruli in response to this odour. The recruitment pattern is odour-specific. However, the responses of the recruited glomeruli decay within 7 minutes after conditioning (Yu et al., 2004).

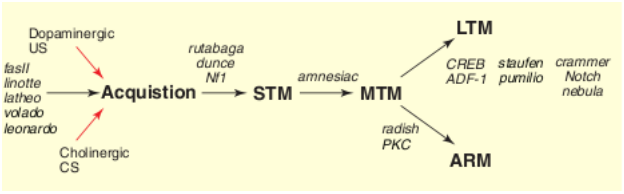
Middle-term memory (MTM) depends on and follows after STM, lasting for a few, at most seven, hours, reaching its peak within the first hour after training. It can be disrupted by anaesthesia and alterations of the *amnesiac*, *dunce* and *rutabaga* genes. It has to be noted that the *amnesiac* gene has its major region of expression not in the MBs but in DPM neurons. The role of DPM neurons will be discussed in 2.3.4. Anaesthesia-resistant memory (ARM) depends on MTM, peaks within 2 hours after training and lasts for no more than 2 days. As its name implies, it can resist anaesthesia. Mutations of the *radish* gene interfere with ARM. Finally, long-term memory (LTM) appears only after spaced and not massed training, peaks within the first day after training and can last for several days. Like ARM, it depends on MTM but it is protein synthesis dependent and sensitive to disruptions of the *dCREB2* transcription factor. One study has suggested that LTM might be localized in the α lobes of MBs (Pascual and Preat, 2001).

2.3.3 Dependence of memory on output from the MBs

Another obviously important issue regards the exact structures where memories are established. Although the involvement of MBs has been confirmed, it can be argued that they simply constitute a preprocessing stage and the possibility of other downstream neurons actually "hosting" the memories cannot be excluded. This issue was addressed by selectively blocking neurotransmission from MBs during the different stages of memory formation and examining exactly when learning performance is affected. The results from experiments of aversive learning



(a) Duration of the different phases (Margulies et al., 2005)



(b) Genetic dissection of Drosophila memory. Dependence of memory on various genes (Margulies et al., 2005)

Figure 2.4: Phases of the fly's memory

indicated that output from MBs is required only during retrieval (testing) of memories and not during acquisition or storage (Dubnau et al., 2001). Another study was even more specific and showed that output from the α/β lobes is required during retrieval (McGuire et al., 2001). These findings are consistent with a model that considers MBs as the memory centre. However, a more recent study cast some doubt on this view (Krashes et al., 2007). A more detailed investigation was carried out which revealed that, for both appetitive and aversive conditioning, neurotransmission from the α'/β' lobes is necessary during acquisition and storage but dispensable for retrieval. On the contrary and in agreement with the previous studies, output from the α/β lobes is required only during retrieval. Therefore, it seems that the MB lobes have a differentiated role with respect to their recruitment during learning.

2.3.4 DPM neurons

DPM neurons have acquired a unique position in the field of insect memory due to the crucial role they play in MTM. It has already been mentioned above that MTM depends on the *amnesiac* gene. However, unlike *dunce* and *rutabaga*, it is not expressed in the MBs. It was discovered that it is strongly expressed in two symmetrical neurons, extrinsic to the MB, the DPM neurons (Waddell et al., 2000). They project exclusively to the MBs and disruption of their function results in behavioural scores similar to the *amnesiac* mutants. Conversely, if the *amn* gene is expressed in DPM cells of *amnesiac* mutants, then memory is rescued.

The link between DPM neurons and MTM is furthered strengthened by the observation

that an MTM trace appears in their activity (Yu et al., 2005). DPM neurons respond both to the US of electric shock and to odours. Their responses to conditioned odours after conditioning do not differ from that to the same odours before conditioning as long as they are examined immediately after odour presentation. However, the responses are markedly increased 30 minutes after conditioning. This memory trace is also lobe-specific since it is observed in the α/α' lobes and, as expected, requires the normal function of the *amnesiac* gene. Considering the above data, it is not surprising that consolidation of MTM requires a prolonged output from the DPM neurons between acquisition and retrieval but not during acquisition or retrieval (Keene et al., 2004), (Keene et al., 2006).

2.3.5 The role of the antennal lobes

The antennal lobes constitute an intermediate stage between sensory neurons and MBs and they are sometimes regarded as a simple relay station. Contrary to this view, several behavioural studies have suggested that they may play a more active role in learning. The activation patterns of AL glomeruli are odour-specific (see subsection 2.2.3). After differential conditioning, these patterns exhibit an increased activity for rewarded odours and decreased for the unrewarded (Faber et al., 1999). In fact, a decorrelation between the rewarded and unrewarded odours was observed (fig. 2.5). Similar results were obtained through direct recordings of neural-ensemble activity in the ALs (Daly et al., 2004). Differential conditioning again leads to the recruitment of more units responding to the reinforced odour. An interesting finding was that some units switched polarity from inhibitory to excitatory and vice versa. Additionally, it was shown that the temporal patterns of activity were significantly affected by conditioning. These data suggest that a restructuring of neural activity takes place at the stage of ALs. The importance of the temporal factor is further stressed by the fact that flies can discriminate even between odours with the same spatial code of AL activation (DasGupta and Waddell, 2008).

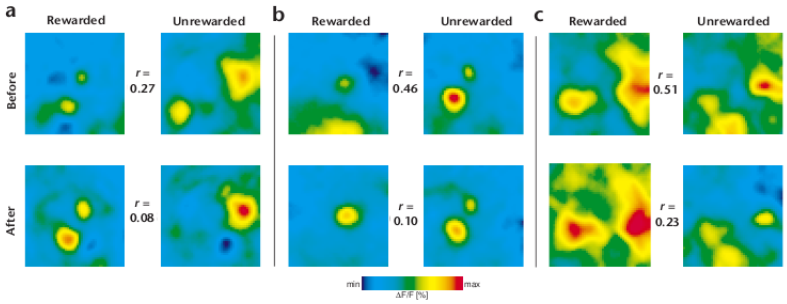


Figure 2.5: Decorrelation of glomeruli activation after conditioning (Faber et al., 1999)

2.3.6 Key points

The purpose of the thesis is not to investigate all the different memory phases and neural processes involved. We are mainly interested in the initial stages of memory formation which is also the base for the later stages. Therefore, our model is going to target STM and the phase of memory acquisition and the MB is going to be treated as a homogeneous structure. More precisely, we could say that we are going to model the α'/β' lobes of the MB (section 2.3.3). For this purpose, it is necessary to include a neuromodulation mechanism which can signal reward or punishment (section 2.3.1). Our value signal in our experiments is going to be only aversive, but, in principle, we make no distinction between aversive and appetitive reinforcement.

2.4 Software models of insect olfactory networks

A complete characterization of the neural olfactory pathway in terms of its neurophysiology and anatomy may still be a goal set for the (perhaps not so near) future but the data gathered so far, as described in the previous sections, have allowed researchers to build models of it at various levels of abstraction. Besides being attempts to reproduce experimental results, these studies have also made predictions and revealed details which were not obvious from the experiments, a sign of success when it comes to modelling. Although the usefulness of such models cannot be doubted, their focus usually is on low-level details of neural circuits without direct references to their behavioural relevance. Therefore, it is time for the appearance of such high-level models. The following sections will present the relatively few existing models.

2.4.1 Models of individual neurons

The PNs and the KCs are the neurons which have received most of the attention since they are assumed to constitute the main channel of information transmission and processing. The peculiar morphology of PNs, having their soma somewhat detached from their dendritic tuft and axon, led to a detailed modelling in order to investigate the interactions between this type of morphology and their electrical properties (Gouwens and Wilson, 2009). Using the NEURON simulation software, a multi-compartmental model was built which included the soma, the axon and the dendritic tuft with its multitude of branches. The model revealed that PNs probably have only one site where spikes initiate and that this spike initiation zone lies at a location distant to the soma. Moreover, there is a need for release sites on many dendritic branches at synapses between ORNs and PNs since a single release site was found inefficient to propagate the input.

Modelling of KCs has not yet included morphological details. Simulations with single-compartmental models, this time using the SNNAP software, were equally insightful though (Wustenberg et al., 2004). It was shown that there is no need to hypothesize the existence

of different subpopulations of KCs since the different spiking characteristics of KCs could be reproduced by manipulating the ratio of g_{Na} to g_K. Investigation of the various current types showed that the sodium current I_{Na} requires two different inactivation time constants implying either that there might exist two different types of I_{Na} or that it goes through more stages than previously assumed. Finally, the existence of a slow potassium current was predicted which can regulate the threshold of the neuron.

2.4.2 Models of networks

Going beyond models of single neurons, the next step that naturally follows is modelling whole networks. As has already been mentioned, the AL can no longer be conceived as a relay station between the ORNs and the MBs, with evidence coming from computational models as well. Even a relatively simple model, using a firing rate formulation, reveals that the dynamics of the AL play an important role in the odour recognition process (Muezzinoglu et al., 2009). This model used a neural network with firing rate neurons to simulate the AL and a support vector machine to classify the output from the AL. Inhibition was found to be important for the decorrelation of odours. A short- term memory trace was also observed in the AL with the sensory transient during odour onset carrying a substantial amount of information. Another study examined the effects of different patterns of interglomerular inhibitory connections and concluded that contrast enhancement is most efficient when such connections are formed between functionally similar (but not necessarily neighbouring) glomeruli (Linster et al., 2005).

More detailed models of the AL have been employed to study the fine temporal patterns of activity that arise in the PNs and LNs. Single-compartmental Hodgkin-Huxley models of PNs and LNs were used to build a model of the AL in order to understand how inhibition between LNs and PNs or LNs and LNs shapes the oscillatory, synchronized PN activity (Bazhenov et al., 2001) (see subsection 2.2.3). The simulation results indicate that PN synchronization relies on fast GABAergic inhibition from LNs whereas the responses of PNs, whose temporal structure is odour-specific, are shaped by LN-LN inhibitory connections. Allowing for some synaptic plasticity in this model can explain a type of short-term memory trace that appears after successive trials (Bazhenov et al., 2005). When the AL encounters an odour for the first time, the LFP does not exhibit any oscillations. In contrast, these oscillations appear only after a few trials, together with a decrease of PN activity. Facilitation of both fast and slow inhibition between LNs and PNs in the model was able to reproduce these experimentally observed phenomena. Interestingly, if excitatory synapses were also allowed to undergo synaptic changes, then the PN representation of odours was found to be noise-resistant. Since this synchronized activity of PNs seems to be used by downstream KCs, a similar study, using again Hodgkin-Huxley models, simulated the behaviour of KCs and the role of feedforward inhibition by LHIs onto KCs (lateral horn interneurons) (Perez-Orive et al., 2004). It was confirmed that KCs act as

coincidence detectors, a property that is due to some active conductances they possess, besides those of sodium and potassium for spike generation. As for the LHIs, the model suggests that their inhibitory drive on KCs acts as a resetting mechanism which limits the time windows of KCs, therefore resulting in their sparse and rare activity and enhancing their coincidence detection characteristics.

2.4.3 Models of the olfactory nervous system

During the last 5 years, a tendency towards more complete models of the whole olfactory pathway has appeared. Of course, the driving force behind such attempts is to understand the neural correlates of the behaviour of insects in this pathway. One of the simplest behaviours is the discrimination of different odours, on the one hand, and the clustering of similar odours in groups on the other. A possible solution to this problem is proposed by a model which divides the KCs population into functional subsets, each associated with a specific LHI (Sivan and Kopell, 2004). The activity of LHIs is assumed to encode the cluster to which an odour belongs whereas KCs provide a parallel pathway with finer discriminatory power. This model was implemented with integrate-and-fire neurons and also showed that oscillatory PN activity is required to account for the ability of fine discrimination.

The next step is to include a learning rule in order to explain more complex behaviours that display the ability of learning. Such a rule, namely spike timing dependent plasticity, is used in a model proposed by Nowotny et al (Nowotny et al., 2005). According to this rule, a synapse is strengthened when a postsynaptic spike follows shortly after the presynaptic spike, within a time window of tens of milliseconds, and is weakened if the timing between the spikes is reversed (Dan and Poo, 2004). Only the synapses between KCs and the extrinsic output neurons of the MB were provided with this plasticity mechanism. After training, the network was able to correctly classify a series of inputs by minimizing the distance between stimuli belonging to the same class and maximizing the distance of stimuli belonging to different classes when the number of input classes was kept under a certain limit. Synaptic plasticity rules which are local by their nature can be combined with more global rules which are believed to have a significant biological relevance in animal learning. This was the approach of a recent model, similar in its architecture to the one just presented above (Huerta and Nowotny, 2009). Using simple McCulloch-Pitts neurons, hebbian learning at its output synapses and a global reinforcing signal to "supervise" the hebbian learning, it could achieve performance comparable to that of SVMs in a recognition task of handwritten digits. Of course, digits are not behaviourally important to insects but their transformation to an "odour-like" representation is straightforward (by pixelating them) and the purpose of the model was not recognition of digits per se but the investigation of its classification capabilities.

Although the above models attempt to simulate a significant portion of the neural pathway

involved in learning, it is still not obvious how they can be directly related to results from actual, behavioural experiments, as described in section 2.3. This issue is addressed by models that explicitly try to replicate some of these results. One such model is shown in fig. 2.6 (Smith et al., 2008). It investigates how and where in the insect brain the CS and the US converge in an associative learning paradigm. The model views KCs as coincident detectors of synchronized, upstream PN activity, with the help of inhibitory drive from LHIs. Short-term learning occurs at the synapses between KCs and lobe neurons (LNs) with a plasticity mechanism based on the interaction between presynaptic activity conveying the CS and neuromodulatory responses from a value neuron (VN) mediating the US. Postsynaptic responses drive longer-term modifications of the synaptic conductances. Pairing PN activity patterns with VN modulatory release was able to induce lasting synaptic modifications that could elicit a specific response from LNs when the "conditioned" CS/PN pattern (but not a partial CS) was later presented as a test stimulus. Another model of a similar architecture but with STDP as the learning rule examined the capabilities for non-elemental associative learning (Wessnitzer et al., 2007). The simulation setup was more similar to actual experimental setups with the activity of PNs being determined by "visual" patterns on a rotating panel. The model could make the association between a punishment (represented as a reflex response) and certain patterns predicting the punishment in negative patterning, biconditional discrimination and feature neutral discrimination paradigms.

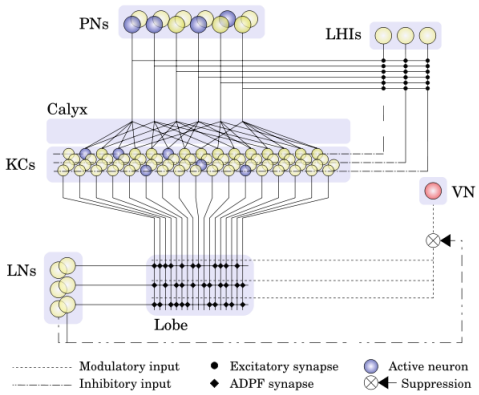


Figure 2.6: Insect brain model by Smith et al (Smith et al., 2008)

2.4.4 Key points

Our explicit goal is to try to link neural processes with observed animal behaviour. Consequently, the finer, low-level details that some of the above studies investigate, like neuron morphology and membrane currents, are not going to be considered, since we do not yet know if and how they might affect behaviour. Our model is going to be built at a higher level of

abstraction. Like most of the models presented in section 2.4.3, we are going to include a plasticity mechanism for the synapses which project from the KCs to the ENs. Additionally, a value signal is going to be used as an indicator of the US.

2.5 Hardware implementations

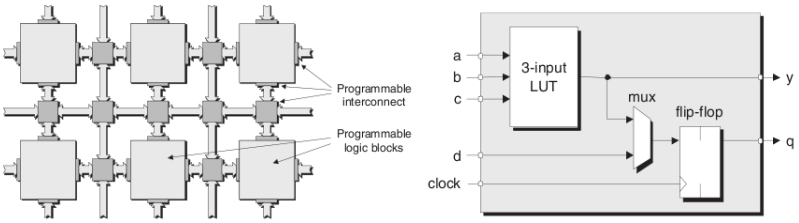
Until now, there have been no attempts to implement the olfactory neural circuits of insects on a hardware platform. In fact, the field of neural networks simulation on hardware is still struggling with the design of more generic networks of a decent size as a proof of concept. No standard tools exist, as is the case on the software side, which give a modeller the ability to specify a network at a more abstract, intuitive level and then simulate it onto hardware. If we want to have any chance of building real-time models of biological neural networks though, then we have to take advantage of the parallelism offered by digital circuits. Just like GPUs take the load of graphics processing off the main processor today, we might be using NPUs (neural processing units) in the future to accelerate simulation of neural networks or achieve real-time performance for robotics applications. In the review of the existing hardware implementations that will follow, we will not cover the cases of microprocessor-based or ASIC solutions as these do not seem suitable for the purposes of the current project. Instead, we will focus on FPGAs.

We will also briefly mention some of the latest developments in the field of neurorobotics, a research methodology which combines (software or hardware) models of neural structures (or even living neural tissues) with a robotic platform in order to more acutely investigate the neural mechanisms of behaviour, according to the principles of embodied cognition (Krichmar, 2008) (Weng et al., 2001) (Seth et al., 2005). Since this methodology has not been employed yet to study (olfactory) learning in insects, we will necessarily examine implementations inspired by other animals.

2.5.1 FPGAs overview

The main feature that distinguishes FPGAs from custom VLSI circuits is the fact that they are “field programmable”. This means that an FPGA board can host different designs and the process of downloading a new design to it takes only a few minutes, with the whole procedure being controlled by appropriate software tools, available directly to the modeller. The basic concept behind an FPGA is depicted in a simplified manner in fig. 2.7. The circuit is composed of an array of independent but interconnected cells. The connections between the cells are not permanent but programmable. In turn, each cell (usually) consists of an LUT (look-up table) which can implement any function of its inputs and a flip-flop (fig. 2.7). Therefore, the cell can function in both a combinatorial and a sequential mode. When a software tool is asked to implement a circuit design, it employs placing, routing and mapping algorithms

in order to determine which logic cells must be used, what functions they should express and how they are to be interconnected. The circuit itself can be designed by using hardware description languages (e.g. VHDL or Verilog) which resemble (but are not) usual programming languages and whose source code can be synthesized into a digital design. Other, even higher level, languages are available as subsets of ANSI C, with certain extensions (e.g. Impulse-C or Handel-C). For a comparison between a VHDL and a Handel-C version of a multi-layer perceptron, see (Ortigosa et al., 2006). The whole process therefore begins with the designer writing his model in a description language which is then passed through a synthesis phase and translated to a digital circuit. Subsequently, this circuit is mapped onto the FPGA elements (a technology-specific stage, in contrast to the synthesis stage) and finally downloaded to the board.



(a) Structure of an FPGA device (Maxfield, 2004)

(b) Contents of an FPGA cell (Maxfield, 2004)

Figure 2.7: FPGA organization

2.5.2 Models of individual neurons

The limited resources of FPGAs constitute a very important constraining factor in any such design, expressed as the percentage of the circuit area or the number of slices/cells/LUTs (two cells usually comprise a slice) that it consumes. Ghani et al (Ghani et al., 2006) proposed a model for an integrate-and-fire neuron that omits multiplication altogether by simply counting the presynaptic spikes until a threshold is reached for the postsynaptic spike to occur. Considering that an FPGA has only a few dedicated multipliers and that recruiting standard logic cells for their implementation is expensive, this approach could conserve valuable resources (it utilized 80 slices for a single neuron with 10 input synapses). On the other extreme, Graas et al (Graas et al., 2004) implemented a detailed Hodgkin-Huxley neuron model (without any synapses though) by using a Simulink library within MATLAB which allows for the design of FPGA projects with blocksets. As expected, this model was much more demanding than the previous one, occupying 44% of the 5120 slices. However, it could “simultaneously” simulate 17 different neurons via pipelining, a kind of time multiplexing whereby the datapath (the computational part of the circuit) is divided in different stages, with buffers in between,

so that each neuron occupies a different stage at each clock cycle waiting to move to the next stage at the next cycle. The same methodology was used to implement a FitzHugh-Nagumo model with a 10 stages deep pipeline (Weinstein and Lee, 2006) and it was later embedded in a more general framework in which the parameters were stored in RAM and constantly fed to the neuron model (Weinstein et al., 2007).

2.5.3 Models of networks

Moving on to the simulation of networks, the most obvious approach is that of a fully parallel design, with each neuron utilizing its own area on the board and buses of signals transferring data among them. This approach was followed by Roggen et al (Roggen et al., 2003) who used a cellular network of 64 neurons to control a Khepera robot in an obstacle avoidance task. The network is called cellular due to its structure, a 2D cell array whose cells can have connections with certain neighbouring cells. The neuron model was a discrete-time version of integrate-and-fire neurons. The design also made use of a softcore processor (a processor that can be instantiated onto the FPGA, just like the rest of the modules), responsible for the communication between the robot and the network. Around 8000 logic cells were required for the complete system, with 1/4 of them consumed by the processor. A biologically more plausible network was implemented by Upegui et al (Upegui et al., 2005), consisting of three layers of 10 neurons each with full intralayer connectivity and full, two-way connectivity between successive layers. A simplified, integrate-and-fire neuron was again used, with presynaptic spike counting as a spike generation mechanism. Additionally, neurons incorporated a type of hebbian learning as well. Almost half of about 2500 slices were needed.

A common theme that seems to be recurring during the last years is the sharing of hardware resources by multiple neuron models. Since the computations required by a neuron module are very demanding in terms of logic cells and dedicated arithmetic units, the implementation of large-scale networks is not possible with fully parallel designs. On the other hand, one such neuron module, with optimizations such as pipelining, can exhibit performance that is orders of magnitude better than real-time. Therefore, a trade-off between speed and area consumption arises. We can sacrifice simulation acceleration in order to gain in network size by recycling multiple neuron models over the same computational units. According to this scheme, the network is divided in subsets of neurons and only neurons belonging to the same subset are updated in parallel whereas the different subsets are updated sequentially.

One design based on this idea was proposed by Ros et al. (Ros et al., 2006). The state of the neurons and the presynaptic events are stored in RAM blocks and are repeatedly fed into the processing units until all neurons are updated. Moreover, each computational unit is pipelined so that 5 neurons per processing unit can be computationally active at each clock cycle. A network of 1024 of integrate-and-fire neurons, partitioned in 4 processing units consumes about

Paper	Neuron model	hw/sw	Neurons No	Syn. No	Slices
(Ghani et al., 2006)	InF	HW	1	10	80
(Graas et al., 2004)	HH	HW	17	0	2250
(Weinstein and Lee, 2006)	FHN	HW	1	0	90-150
(Weinstein et al., 2007)	HH	HW	40	1.600	13840
(Roggen et al., 2003)	InF	HW	64	1.664	8000
(Upegui et al., 2005)	InF	HW	30	900	1250
(Ros et al., 2006)	InF	hybrid	1024	-	20000
(Maguire et al., 2007)	InF	hybrid	4200	1.964.200	-
(Pearson et al., 2007)	InF	HW	1120	9.120	-

Table 2.1: Summary of hardware implementations of biological neural networks. InF=Integrate-n-fire, HH=Hodgkin-Huxley, FHN=FitzHugh-Nagumo.

20000 slices. It has to be noted though that this was a hybrid hardware-software design, using the FPGA to update the membrane potentials and the CPU for learning and communicating spikes between neurons. The same idea was implemented with a different approach by Maguire et al. (Maguire et al., 2007). The time multiplexing was controlled by softcore processors (Xilinx Microblaze in this case) which handled the transfer of neuron/synapse states from RAM to computational units and then back to RAM. It was possible to simulate a total of 4200 neurons with 4 Microblaze processors. Pearson et al (Pearson et al., 2007) designed a neuroprocessor implemented fully on hardware, without the support of any processors, following the same idea. Their model consisted of 10 neural processing elements, each being responsible for updating 112 neurons and 912 synapses, stored in RAM. A sequencer was in charge of communication and coordination between the processing elements. Table 2.1 presents a summary of the hardware implementations described above and their requirements.

2.5.4 Neurorobotic studies of learning and memory

The field of neurorobotics is quite new, something which should not come as a surprise, considering the significant technical work required and the increased demand for computational power by the simulated models of biological neural networks. The task of building a neurorobotic platform seems less and less daunting as more computational power and simpler robotic kits become available. The community of neuroroboticists finds itself in a growing phase and has presented, during the last 10-20 years, some very interesting examples of how animats can help us to understand animals.

One of the most well-known neurorobotic platforms is the series of Darwin robots (Edel-

man, 2007). They are usually equipped with cameras, microphones, IR and other sensors and their motor system is controlled by a simulated nervous system. The simulation follows the so-called synthetic neural modelling approach, running on Beowulf clusters, with models which contain tens of thousands of neuronal units (not single neurons but neural masses of about 100 neurons, represented by their mean firing rate), inspired mostly by the rat brain. It is a setup which allows for a comprehensive recording (and therefore later processing) of the robobrain's activity during the experiments, something which is not feasible with living animals. The different versions of Darwin (currently version XI) have investigated various aspects of (rat) brain functioning, like vision (Almassy et al., 1998), the role of hippocampus in spatial memory and maze navigation (Krichmar et al., 2005) (Fleischer et al., 2007), perceptual categorization and conditioning (Krichmar and Edelman, 2002).

Besides Darwin, the rat hippocampus and its place cells have attracted the attention of many other research groups as well, trying to understand how rats create an "internal map" of their surroundings (Arleo et al., 2004) (Banquet et al., 2005) (Milford et al., 2004) (Cuperlier et al., 2007). The basal ganglia are another group of structures that have been embedded on a robot and studied, with respect to their role as an action selection mechanism (Prescott et al., 2006), whereas another study investigated the role of neuromodulation on decision making (Sporns and Alexander, 2002). Moving beyond experiments with single robots, some research groups have started employing ideas of evolution whereby multiple robotic agents can interact and exchange "genetic material" in order to improve the efficiency of their neural networks (Floreano and Keller, 2010) (Doya and Uchibe, 2005).

Unfortunately, the invertebrate-based neurorobotic studies on learning are still limited. For example, Damper et al (Damper et al., 2000) have developed a robot model of *Aplysia* in order to investigate both simple forms of learning, like habituation and sensitisation, and higher-order conditioning. On the other hand, other aspects of insect and invertebrate behaviour have received more attention, e.g. visual and auditory processing (Franceschini, 2008), (Horchler et al., 2004). A large number of studies have also examined the neural mechanisms of walking, navigation etc. (for reviews, see (Webb, 2002), (Pfeifer et al., 2007), (Ijspeert, 2008)).

2.5.5 Key points

It seems a bit surprising that FPGAs have not been more extensively used in neurorobotic studies. The technology of FPGAs is now powerful enough to allow us to model neural networks of a substantial size and opens up new possibilities for research in the field. Our robotic platform moves towards this direction.

Due to the fact that our hardware simulator will be required to model networks of various sizes and connectivity patterns, it should be obvious from the above discussion that we are going to use the idea of sharing hardware resources as well. The state of the network is going

to be stored in RAM blocks and a limited number of computational units should be responsible for updating it. In order to make our simulator as technology-independent as possible, it is going to be a purely hardware design. Communication with the "external" world should be restricted to receiving the necessary input patterns.

Chapter 3

Implementing biological neural networks on FPGAs

FPGAs are far from being the preferred simulation platform for most computational neuroscientists and there is a good reason for this reluctance. Until quite recently, developing electronic systems directly on hardware was confined only to a few R&D groups in the academia or in the semiconductors industry. With the advent of FPGAs, a wider user base can now conduct such research. Moreover, they have allowed for a much greater flexibility during the design process which can substantially facilitate rapid prototyping, a crucial factor in a field like computational neuroscience where there is great uncertainty as to which parameters, models and neural architectures are the most important. However, the community has been reluctant to catch up with the new technology. Working with hardware seems like a daunting task to a traditional computer scientist with limited exposure to digital design techniques and is simply out of the question for a biologist struggling to run a MATLAB script. In this chapter, a tool will be presented that gives a researcher the ability to move quickly from a high-level description of a neural network to its hardware equivalent, ready to be run on an FPGA.

3.1 Wetware

When designing a neural network, certain decisions have to be made as far as the appropriate level of simulation is concerned. This is especially true when it comes to biological neural networks for which achieving a predefined goal or performance is not necessarily of the highest priority. The network needs to have some biological relevance if it is to be useful at all in providing us with some insight regarding the link of certain neurobiological processes to behaviour. Choosing the right abstraction level is still a hotly debated subject, as the recent,

SyNAPSE vs BlueBrain cat fight shows [1].

What is the criterion for this choice and what exactly is our frame of reference for a "right" abstraction level? The field of biorobotics can be of significant help on this issue. It is hard to imagine where this, increasingly "esoteristic", process of including more and more details into a model can stop without having an "external" reference point, as is usual for engineers who model physical systems. The actual behaviour of animals (and animats) may serve as a solid reference point. Therefore, besides using what is generally known and accepted about the wetware of the insect brain in order to build a robot with learning capabilities, this thesis will also attempt to make an initial step towards the inverse direction, that is, to examine the validity of these assumptions by testing their function in a real environment.

Of course, it is impossible (and probably not even desirable) to include each and every aspect of the underlying neurobiology in our model, at least not at this initial phase of the project. We'll have to contend ourselves with a preliminary core of basic features that seem to be the most behaviourally relevant, according to the current state of the research. In what follows in this section, the choices regarding the neurobiological properties to be implemented will be described and justified. It has to be noted that these choices are not determined only by theoretical questions such as the above but are decisively constrained by hardware requirements imposed by the technology of FPGAs as well as by the requirement for real-time performance in the case of robotic experiments.

3.1.1 Neuron and synapse models

As is usual in models of neurobiological structures, the individual neuron is going to be the basic unit for our models as well. This is not an obvious choice, as it may seem, since there have appeared attempts to model the brain at a more mesoscopic level, using neural masses as the fundamental building blocks (Freeman, 2008), (Freeman, 1975). However, the results from such models are quite difficult to interpret and there is a considerable lack of experimental data to drive a modelling attempt, at least as far as the insect brain is concerned.

Neuronal modelling is a whole field in itself and there is a great variety of available models (Dayan and Abbott, 2001). A first dilemma is whether to use single-compartment or more detailed, multi-compartment models. The latter choice, with respect to the goals of our research project, would serve only to provide us with information that we would not know how to handle since there are no experimental data linking neuron morphology with learning behaviour in insects, although it would be an interesting research path to follow in the future. Therefore, single-compartment models will be preferred.

From the simple integrate-and-fire to the more realistic Hodgkin-Huxley models, there is

[1] See http://spectrum.ieee.org/tech-talk/semiconductors/devices/blue-brain-project-leader-angry-about-cat-brain for some objections with regard to SyNAPSE, raised by BlueBrain's leader scientist.

a trade-off for each model between its biological plausibility and its computational efficiency. A model proposed recently by Izhikevich (Izhikevich, 2003) manages to strike a nice balance between these two requirements. While it can reproduce the firing patterns of many types of neurons, with the right choice of the parameters, it requires just 13 operations per 1 ms of simulation, comparing favourably with other models (Izhikevich, 2004). It is a phenomenological model which does not describe the activity of any membrane channels. Nevertheless, it can exhibit a varied range of behaviours.

Communication between neurons is achieved of course through synaptic transmission. The input that a neuron receives from a synapse is determined by the synaptic conductance and the neurotransmitter concentration at the synaptic cleft. Consequently, the most basic variable that a synaptic model should compute is that of the neurotransmitter concentration. This whole process is relatively easy to model since it mostly depends on the presynaptic spikes arriving at the synapse which force the neurotransmitter concentration to rise sharply, following afterwards an exponential decay (Dayan and Abbott, 2001), (Destexhe et al., 1994).

Therefore, the neural network will be governed by the following equations :

$$C\frac{dv}{dt} = k(v - v_r)(v - v_t) - u + I \tag{3.1}$$

$$\frac{du}{dt} = a(b(v - v_r) - u) \tag{3.2}$$

$$\text{if } v \geq 30 \text{ mV, then} \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \tag{3.3}$$

$$\frac{dp_s}{dt} = -p_s \frac{1}{t_s} \tag{3.4}$$

$$dp_s = p_{max}(1 - p_s), \text{ if presynaptic spike} \tag{3.5}$$

$$i = p_s \cdot g_{max} \cdot (v_{rev} - v) \tag{3.6}$$

where v is the neuron membrane potential, u the recovery variable, v_r the resting potential, v_t the threshold potential, a, b, c and d parameters of the Izhikevich model, I the total input, p_s the neurotransmitter concentration, t_s the time constant of its decay, p_{max} the maximum neurotransmitter concentration, v_{rev} the reversal potential, g_{max} the synaptic conductance and i the input from this specific synapse ($I = \sum i$). More accurately, p_s is not exactly the neurotransmitter concentration but the probability of a postsynaptic channel being open or the percentage of open, postsynaptic channels due to the presence of neurotransmitter and g_{max} is the synaptic conductance when all postsynaptic channels have opened.

3.1.2 Learning mechanisms

Since Hebb proposed his simple learning rule, research on plasticity mechanisms has largely focused on how modifications of the synaptic strength can affect memory and learning. It is widely accepted that the engram is to be found in the synapses and the way they react to pre- and post-synaptic activity, although other mechanisms, such as changes in neuronal intrinsic excitability (Zhang and Linden, 2003), have been put forward as well. However, since these, possibly complementary mechanisms, have not yet been extensively investigated, they will not be included in our model.

The simple Hebb rule can lead to learning that suffers from problems of instability and loss of selectivity since it can result in uncontrolled and homogeneous synaptic growth (Dayan and Abbott, 2001). Several modifications have been proposed that can impose constraints of synaptic saturation and competition in order to address these problems. However, many of them are not biologically plausible. Spike timing-dependent plasticity (STDP) seems to be a biologically realistic type of learning that can achieve both stability and competition (Song et al., 2000b).

Until recently, there had been evidence for the presence of STDP in biological systems (Dan and Poo, 2004), (Roberts and Bell, 2002) but it hadn't been observed in invertebrates. A "fortuitous observation" during in vivo recordings in the locust brain (Cassenaer and Laurent, 2007) gave rise to this possibility when a spontaneous action potential of a β lobe neuron (β-LN) after stimulation of its presynaptic KC greatly enhanced the following EPSP. A series of experiments confirmed that these synapses actually do behave in an STDP manner. It is important to note that, in those experiments, all the stimuli were artificial. Therefore, it is not clear how the observed STDP could be behaviourally relevant as far as learning and memory are concerned. In fact, a later study (Ito et al., 2008), involving behavioural experiments, showed that the relation of STDP to learning is not at all straightforward.

STDP is a type of Hebbian learning which depends critically on the temporal order of pre- and post-synaptic spiking. According to STDP, an increase in synaptic strength is induced when a presynaptic spike is quickly followed by a postsynaptic whereas synaptic depression occurs when the order is reversed and the postsynaptic spike precedes the presynaptic. The time window within which STDP can be observed is in the order of tens of milliseconds. STDP can be modelled by the following equation (Song et al., 2000b) :

$$F(\Delta t) = \begin{cases} A_+ \exp\left(\frac{\Delta t}{\tau_+}\right) & \text{if } \Delta t < 0 \\ -A_- \exp\left(-\frac{\Delta t}{\tau_-}\right) & \text{if } \Delta t \geq 0 \end{cases} \tag{3.7}$$

The function $F(\Delta t)$ determines the amount of synaptic modification. The parameters A determine the maximum amounts of synaptic modification and the parameters τ the time windows (pre- to post-synaptic inter-spike intervals) over which STDP can occur.

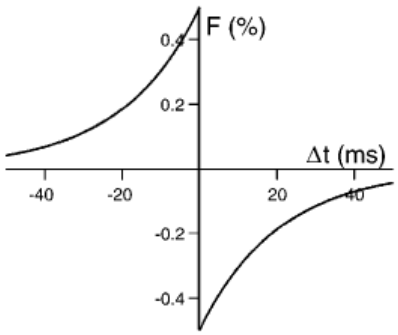


Figure 3.1: The STDP window (modification function) (Song et al., 2000a)

In order to implement STDP in a computational model, each synapse needs to have two functions, $M(t)$ and $P_a(t)$ (Song et al., 2000b). $M(t)$ is used to decrease the synaptic strength whereas $P_a(t)$ is used to increase it. $M(t)$ is decreased by a certain amount A_- whenever the postsynaptic neuron fires and, if this is followed later by a presynaptic spike, then the conductance is decreased by $M(t)g_{max}^{stdp}$, where g_{max}^{stdp} is the maximum value that the synaptic conductance can acquire through STDP modification, i.e. if the conductance exceeds at some point this value, then it is reset to it. $P_a(t)$ is increased by A_+ each time the presynaptic neuron fires and, in case a postsynaptic spike occurs shortly afterwards, the conductance is increased by $P_a(t)g_{max}^{stdp}$. Both $M(t)$ and $P_a(t)$ decay exponentially when there is no spiking activity. The required equations are the following :

$$\tau_- \frac{dM}{dt} = -M \tag{3.8}$$

$$M \rightarrow M - A_-, \text{ if postsynaptic spike} \tag{3.9}$$

$$g_{max} \rightarrow g_{max} + M(t)g_{max}^{stdp}, \text{ if presynaptic spike} \tag{3.10}$$

$$\tau_+ \frac{dP_a}{dt} = -P_a \tag{3.11}$$

$$P_a \rightarrow P_a + A_+, \text{ if presynaptic spike} \tag{3.12}$$

$$g_{max} \rightarrow g_{max} + P_a(t)g_{max}^{stdp}, \text{ if postsynaptic spike} \tag{3.13}$$

$$g_{max} \rightarrow \begin{cases} 0 & \text{if } g_{max} < 0 \\ g_{max}^{stdp} & \text{if } g_{max} > g_{max}^{stdp} \end{cases} \tag{3.14}$$

Besides STDP, another plasticity mechanism which seems to be quite significant for the insect brain is that of neuromodulation (Schwaerzel et al., 2003). Neurons that secrete neuromodulators, such as dopamine and octopamine, are considered to mediate the signals of reward and punishment in experiments of associative learning. It is still not exactly known how neuromodulators affect the process of synaptic strengthening or weakening. Nonetheless, since neuromodulators play such an important role in learning, an initial attempt will be made to include this mechanism in our model, with the understanding that it is based on certain, possibly invalid assumptions.

Inspired by (Izhikevich, 2007), another variable is added (c_{tag}) for each synapse, which acts as a synaptic tag. This variable follows the STDP rule but does not affect directly the synaptic conductance. Instead, it simply "tags" a synapse as a candidate for modification, in case a neuromodulator is released from a neuron that targets this specific synapse. The neurons which release neuromodulators are modelled as simple triggers i.e. they "spike" when externally stimulated, with the concentration of the neuromodulator (c_{mod}) decaying exponentially afterwards. The change in conductance depends on both the synaptic tag and the neuromodulator concentration. The following equations describe the above process in mathematical terms :

$$\frac{dc_{tag}}{dt} = -c_{tag}\frac{1}{t_{ctag}} \tag{3.15}$$

$$c_{tag} \rightarrow \begin{cases} c_{tag} + M & \text{if presynaptic spike} \\ c_{tag} + P_a & \text{if postsynaptic spike} \end{cases} \tag{3.16}$$

$$\frac{dc_{mod}}{dt} = -c_{mod}\frac{1}{t_{mod}} \tag{3.17}$$

$$c_{mod} \rightarrow c_{mod} + c_{mod}^{max}(1 - c_{mod}), \text{ if trigger} \tag{3.18}$$

$$\frac{dg_{max}}{dt} = c_{tag}c_{mod} \tag{3.19}$$

3.2 Hardware

It is not uncommon for simulations of biological neural networks to take hours or even days to complete, even for networks of relatively small size. In order to get acceptable performance, at least for large scale simulations, clusters have to be employed, using expensive hardware (see, for example, the supercomputer used for the Blue Brain project, with its 4096 nodes (Markram, 2006)). This approach has also been followed in certain cases where the neural network is used to guide the behaviour of a robot (Edelman, 2007), (Krichmar et al., 2005). However, this

seems to be a contradictio in terminis since autonomy is one the features that make robots what they are. If we want robots that can behave more autonomously, then we need solutions which are more "portable". In this section, the capabilities and limits of one such solution will be investigated.

The requirements that our hardware simulator should be able to meet are the following :

- It should be able to simulate networks described by the equations of section 3.1, with neurons following the Izhikevich model and with synapses that could be fixed or plastic, following either the STDP learning rule or the neuromodulation rule. There should be no constraints as far as the connectivity pattern is concerned.
- With regard to the size of the network, it should have the ability to simulate networks whose size is similar to that of the fly's olfactory pathway (around 150 PNs and 2500 KCs, with $\frac{No\ of\ synapses}{No\ of\ neurons} \approx 10$).
- The simulator should be fast enough for robotic experiments, i.e. it should achieve real-time (or at least near real-time) speed performance.

3.2.1 Architecture of the simulator

Is it possible to build a "portable" cluster? Should we build one and what parts of a processor are necessary in this case? Should we even insist on the idea of using a processor or could we get increased performance if we move towards more direct hardware implementations? These are engineering questions which would be easier to answer if we had clearly defined requirements for our design. Of course, requirements analysis is a standard procedure which every software or hardware system has to go through, with great attention. The above discussion on the wetware of the insect brain could be considered as some kind of such an analysis. It should be obvious, though, that computational neuroscience, especially that branch which attempts to draw links between neurobiological data and behaviour, still stands at an exploration phase where certain assumptions might prove wrong and proposed mechanisms might be discarded as irrelevant to learning in the future, e.g. neuronal intrinsic excitability might become more important as a learning mechanism than modification of synaptic weights.

If we look more closely, with the eyes of an engineer and not those of a neuroscientist, we can discern that behind all this talk about membrane potentials, synaptic clefts and neurotransmitters, what remains is mainly a series of differential equations which update certain variables of interest with each simulation step. No matter what mechanisms and models we include, it is expected that they will also follow this pattern of the need to massively update elements which follow the same computational path. Therefore, we could focus on this super-problem and be relatively confident that the design could be easily amended, in case we are required to add another mechanism or substitute one.

Of course, the most efficient way to solve the differential equations of our system would be to use for each individual element (in our case, for each neuron and each synapse) dedicated hardware resources so that all variables are updated simultaneously (see 2.5.3) and direct connections for communication among the elements. This approach may work for small networks of well-patterned connectivity, but it is impossible to scale, since it quickly consumes all hardware resources and the design would be extremely hard to route for more complex connectivity patterns. Consequently, the idea of sharing hardware resources and using RAM memory to store the network state comes into view.

There still remains the issue of how the necessary arithmetic operations should be performed. We could produce the new values from the equations by "unrolling" them and breaking them down into their main operations. For example, a simple equation, like $(a+b)*c$, can be mapped to an adder, with a and b as its inputs, followed by a multiplier, operating on c and the result of the adder. If we introduce buffers between the different operations, we could also pipeline this datapath (see 2.5). Although it could lead to very fast implementations, the downside of this approach is that the datapath is fixed, suitable only for a specific equation. Therefore, we would have to implement one such datapath for each of the 19 different equations of our model (see 3.1), with many of them remaining idle while waiting for the longer computational paths to complete. In addition to that, a new datapath has to be included for every new equation, e.g. in case we decide to use a different neuron model. For these reasons, a more flexible solution would be preferable for the purposes of this project.

3.2.1.1 A network of cores

In order to take advantage of the parallelism offered by the FPGA hardware, the simulator has been designed in the form a network of cores. The cores can work in parallel and each of them is responsible for updating a specific part of the neural network. They are composed of a computational unit, whose task is to perform all the necessary computations and a RAM memory which holds the state of the neural network (the part of the network for which this unit is responsible). A communication module is also included so that the spikes from one part of the network may be transmitted to the other parts. The idea is that we should be able to improve the performance of the design simply by adding more cores to it, as long as there are available hardware resources.

In fig. 3.2, a block diagram of the whole network is depicted. The FSM's function is to allow communication of the network with the external world, i.e. receive external input to certain neurons and write it to the appropriate RAM addresses and also read RAM addresses and present their contents to the appropriate output port. This is the point of interaction between software (drivers) and hardware. The VHDL source code which implements the network of fig. 3.2 has been written in such a way, using *generate* statements, so that the number of cores is a

simple parameter in a configuration file and can be easily modified.

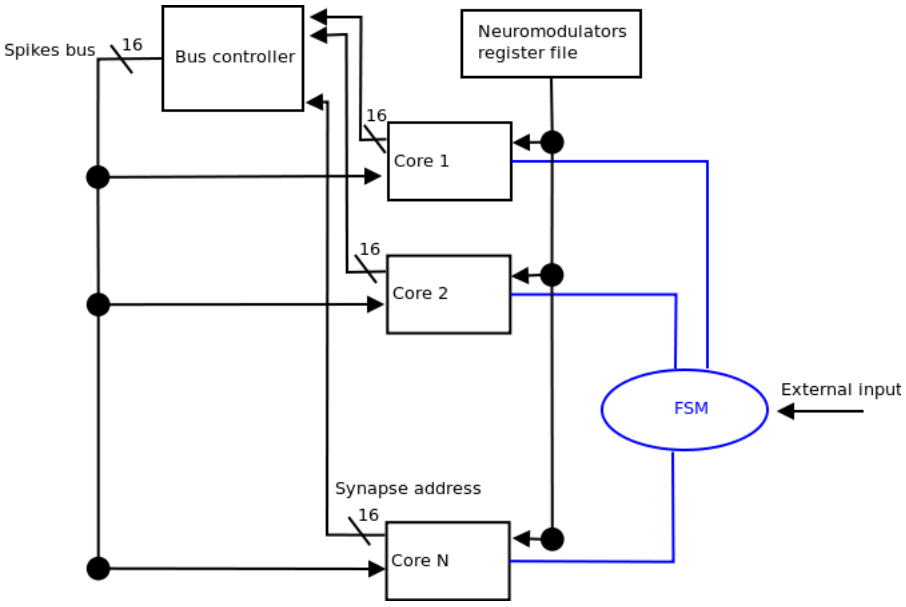


Figure 3.2: The hardware simulator as a network of cores

3.2.1.2 Memory organization

Fig. 3.3 depicts the way RAM memories are organized within each core. A RAM block has a width of 16 bits and is initially segregated into three different sections, the first of which corresponds to the neurons of the neural network, the second to the synapses and the third to the neuromodulators (if there are any). According to the Izhikevich neuron model, the state of each neuron is determined by its membrane potential and its recovery variable. Moreover, the input current is required in order to solve equation 3.1. These are the three variables that need to be stored in RAM for each neuron. There is something that seems to be missing, though. The parameters of the Izhikevich model (a, b, c, d, C_m, k, dt). In an attempt to eliminate unnecessary repetitions, parameters like these and like the ones used by the equations for the synapses and learning mechanisms, do not accompany each neuron (or each synapse). Instead, each neuron is also characterized by its type (see 4th entry in fig. 3.3), meaning a set of values for the parameters, defined as constants in VHDL configuration files, so that the values of the parameters can be retrieved according to the neuron type. The rest of the memory slots dedicated to a neuron are pointers. Most of them are pointers to the output synapses of the neuron, essentially defining the connectivity pattern of the network, used whenever the neuron generates a spike. Another pointer is used to indicate the group of the input synapses to the neuron which provide the input current and another one which points to the next neuron in the neurons' section.

The synapses' section is a bit more complicated. Although the neurons are ordered sequentially, the synapses follow a different ordering pattern whereby they are grouped together according to the neuron to which they project. The synapses' section begins with the first synapse projecting to the first neuron (the first input synapse of the first neuron), followed by the second input synapse and the rest of the input synapses. The second group of synapses begins with the first input synapse of the second neuron etc. This organization was chosen in order to facilitate the computation of the input currents. Each group has a pointer to the postsynaptic neuron and its membrane potential, which is required for the computation of the synaptic currents (equation 3.6). As the synapses of a group are being updated, their synaptic current is added to an accumulator and, upon completion of the group update, its value is stored back to the slot indicated by the postsynaptic neuron pointer. The presence or absence of a learning mechanism, as well as its type, introduce another complication. Regardless of whether a synapse is plastic or not, the neurotransmitter concentration is a variable always needed and, as a result, always included in RAM for every synapse. On the other hand, the other synapse variables (conductance, STDP variables M and P_a, synaptic tag), shown in fig. 3.3, are not always required. In fact, usually most synapses of a network have fixed conductances with only a relatively small subset having learning capabilities. Therefore, the conductance and STDP variables M and P_a are stored in RAM only for those synapses which follow the STDP or the neuromodulation learning mechanism whereas the synaptic tag is stored only for those whose plasticity depends on neuromodulators (see relevant equations in 3.1.2).

It should be obvious from the above discussion that the memory organization pattern of the synapses renders the factor of fan-in much more important than that of fan-out. Whereas each extra output synapse simply requires another pointer to be added, input synapses are accompanied by a significant number of relevant variables. This means that the update cycle for neurons with high fan-in is more costly, in terms of computation time, a fact which should be taken into account when assigning neurons to the various cores (see also discussion about load distribution in 3.3.3). Moreover, the fact that a neuron has to carry with it all of its input synapses and that the neurons are ordered in a serial manner makes it possible for the RAM blocks to be unevenly occupied. In case there are consecutive neurons with high fan-in, a substantial portion of the RAM blocks hosting these neurons might be left unutilized.

Finally, the neuromodulators' section is the simplest of all. For each neuromodulator, we need just two memory slots. One of them holds the id and the type of the neuromodulator and the other one the neuromodulator concentration. A neuromodulator affects only those synapses whose plasticity mechanism (first entry for each synapse in the RAM block) is equal to the id of this neuromodulator. If all bits of the plasticity mechanism are 0s, then the synapse is fixed, whereas if they are all 1s, the simple STDP is used. All the other combinations of bits can be used to point to a neuromodulator.

It should be noted that the RAMs used in this design correspond to block RAMs, residing inside the FPGA chip which the synthesis tools infers from the VHDL source code. This allows for a more direct, self-contained and technology independent design but it can impose serious constraints on the size of the networks that can be implemented, especially in the case of FGPA boards with limited hardware resources.

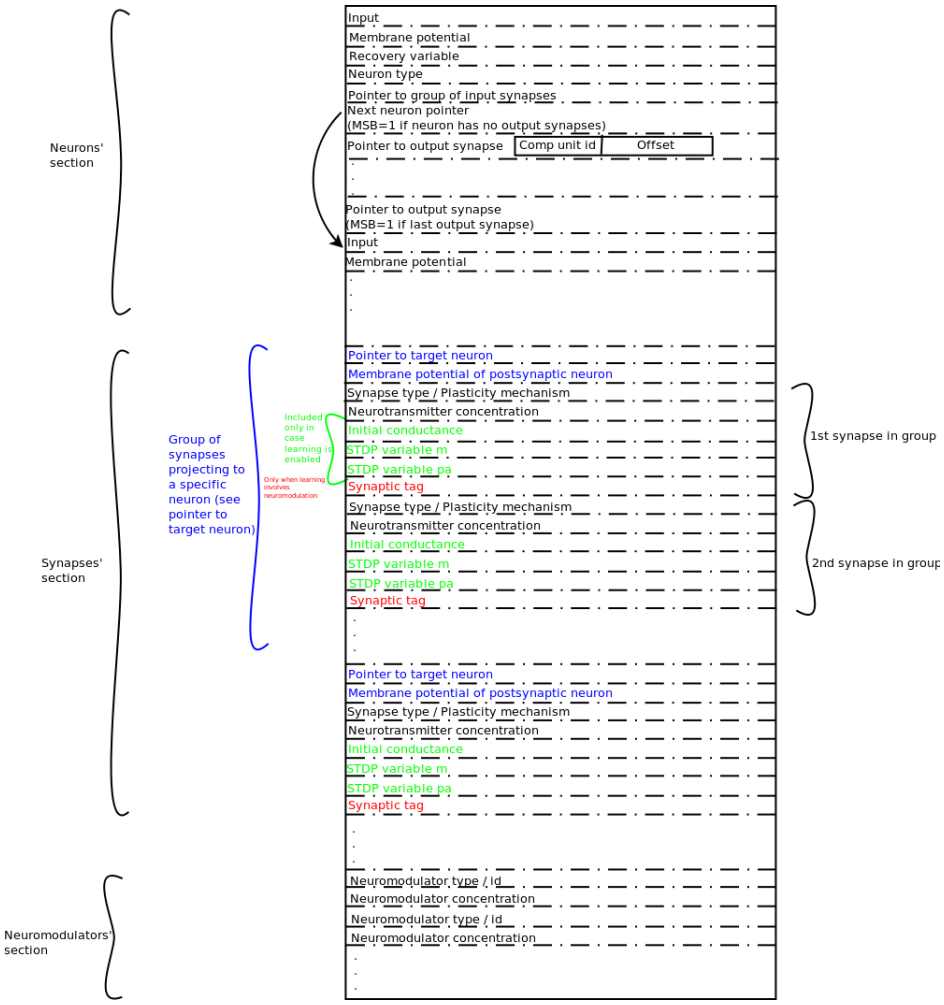


Figure 3.3: Organization of the RAM blocks

3.2.1.3 Computational units

The basic unit that performs all the arithmetic calculations is shown in fig. 3.4. It is composed of a register file, an arithmetic unit, multiplexers which select the inputs to the arithmetic unit and a finite state machine (FSM). The register file is used to hold the variables, as they are fed to the unit (for example, the membrane potential of a neuron), the necessary parameters (e.g. the membrane capacitance, according to the neuron type) and some intermediate, temporary results from the arithmetic unit.

The main function of the FSM is to go through all the computation steps in order to update the value of a variable by selecting at each cycle the correct inputs and operation for the arithmetic unit and subsequently storing the result in the register file or using it for the next step. For example, in order to compute the new value for the neurotransmitter concentration (eq. 3.4) according to the Euler method, we would have to follow the computational path : $p_s^{new} = p_s^{old} - dt * \frac{p_s^{old}}{t_s}$ or $\frac{p_s^{old}}{t_s} \rightarrow result * dt \rightarrow p_s^{old} - result$. First, the FSM selects the old value of p_s and the time constant (t_s) as inputs to the unit from the register file and division for the operation. The result is then used again for the next computation, a multiplication by the time step dt and this result is subtracted from the old value of p_s.

Since division is expensive, it has not been included in the operations which the arithmetic unit can perform. Instead, the inverse of the parameter t_s (and every parameter which acts as a divisor) is used in a multiplication. Fixed point 32 bit arithmetic is used, with 16 bits devoted to the fractional part. Of course, the values stored in RAM are 16 bits wide and they have to be converted to 32 bits before being used by the arithmetic unit.

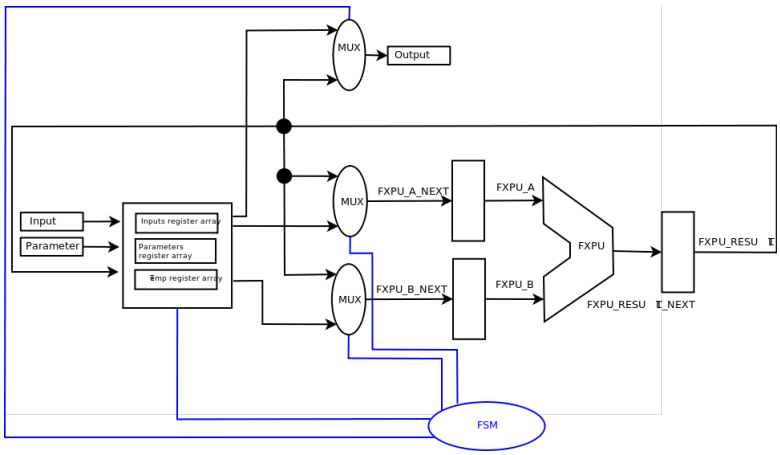


Figure 3.4: Block diagram of a computational unit

3.2.1.4 Communication among the cores and between the RAM blocks and the computational units

The computational units do not have direct access to the RAM blocks. They can only read the variables that they need from an input FIFO (first in, first out) structure, execute the computational path and write the new values to an output FIFO. Another structure is responsible for the communication between RAMs and computational units (fig. 3.5). This communication module constantly checks the status of the input and output FIFOs. In case the input FIFO has enough empty slots to accept data for a neuron, synapse or neuromodulator (depending on the RAM section that is active), it uses a pointer to read this data from the RAM block and push it

to the FIFO. For example, when updating the neurons' section and the input FIFO has four or more empty slots, then the neuron type, input current, membrane potential and recovery variable of the neuron pointed by a "read pointer" can be read and pushed to the FIFO. Likewise, a "write pointer" is used, when the output FIFO holds new data, in order to update the RAM memory with the new values.

Another crucial function of this module is the transmission and reception of spikes, i.e. communication between the different cores. As mentioned above, the synapses that project to the same neuron are grouped together in the RAM blocks. In addition to that, when a neuron is assigned to a specific RAM block during the creation of the memory initialization files, the group of its input synapses is also assigned to the same RAM block, thereby avoiding the need to communicate the postsynaptic membrane potentials to other cores. Only spikes need to be transmitted, since the output synapses of a neuron may be hosted in the RAM block of another core. When the communication module detects that a neuron has generated a spike, it starts reading the addresses of this neuron's output synapses and passes them over to the spikes' bus after having requested and been given access to it (fig. 3.2). The first part of the address of an output synapse consists of those bits that point to the core where the synapse is located whereas the least significant bits comprise the offset inside the RAM block of this core. As an example, if a neuron has an output synapse which is located at address 32 of core no 3, then the address of this synapse should be "0011000000100000", if we use the four most significant bits for core addressing and the twelve least significant for the offset.

Each core has a FSM which listens to the spikes' bus. Every time a new address is being driven to the bus, each of these FSMs checks the most significant bits of the bus in order to determine which core this synapse belongs to. If it belongs to the same core as the FSM, then it gets registered into a spikes' FIFO until the RAM memory is freed, at which time it can be transferred to it. By "transferring" the spike to the RAM block, it is simply meant that the spike bit of the synapse (stored in the same address as the synapse's type and learning mechanism, fig. 3.3) is set to 1. Of course, using a simple bus to transmit spikes can become quite inefficient when we have many cores, generating many spikes, trying to access the bus at the same time and waiting for each other. The advantage is that it is relatively simple to implement and cheap, in terms of required slices. In a future redesign, it should be replaced by a more "intelligent" mechanism.

Besides spikes, there is also something else that a core might need, not residing in its RAM block. In case there are synapses whose conductance is affected by neuromodulators, then the concentration of this neuromodulator is required in order to compute the new value for the conductance. However, the values of neuromodulator concentrations are not transmitted, like spikes, from the cores which have them to those that need them. In order to avoid this communication overhead, another solution was preferred. Considering that there is a limited number

of neuromodulators, it is more efficient to use a register file to store the neuromodulators' concentrations, in addition to the RAM section dedicated to them (fig. 3.2). Each core can have direct access to this register file, provided that another core is not attempting to access it at the same time. After the neuromodulators' RAM section has been updated, the new values are also copied to the register file so that they can be used by synapses during the next update cycle.

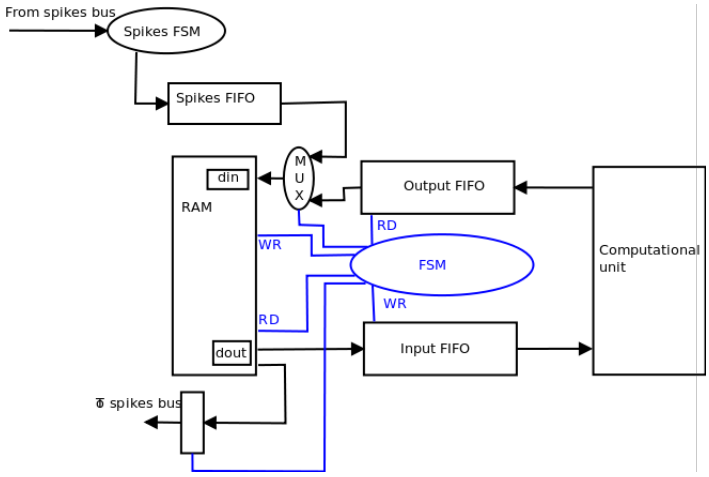


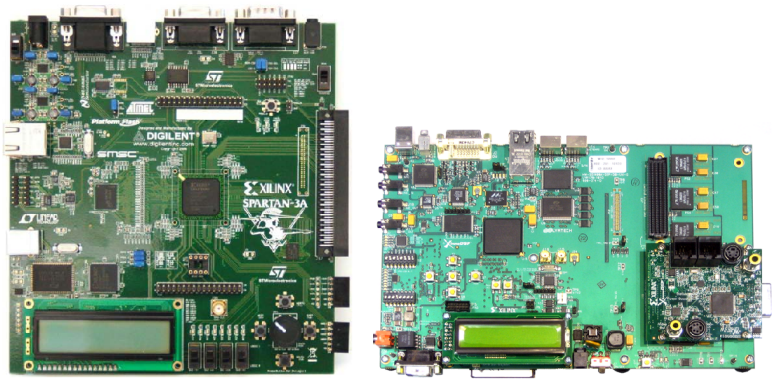
Figure 3.5: Block diagram of a core's communication module

3.2.2 Integrating the network within an embedded system

Since the aim of this project is to use the design described above in order to simulate a neural network that can guide the behaviour of a robot, we can take advantage of the availability of soft-core processors (processors which are not "hardwired" but can be configured and downloaded to the FPGA, just like other designs) and build an embedded system in which the neural network can be a peripheral, attached to the processor via a bus. An embedded system can greatly simplify the communication of the FPGA with the robot. We also need to bear in mind that the mechanisms under investigation are those of learning and memory and not those of either sensing or motor control. Therefore, the neural network cannot assume full control of the robot, from the sensing stage to the final motor decisions (more on this in chapter 4). There is a need both for pre-processing, before the sensory input is fed to the network, and for post-processing that "interprets" the output of the network, tasks much easier to accomplish in software than hardware.

The embedded system was built with the two boards, shown in fig. 3.6. The FPGA of the Xilinx Spartan-3A XC3S700A board has 13.248 logic cells (5.888 slices), 360K bits of block RAM (20 blocks) and 20 dedicated multipliers (Xilinx, 2007). On the other hand, the XC3SD3400A FPGA is more powerful, with 53.712 logic cells (23.872 slices), 2268K bits of

block RAM (126 blocks) and 126 DSP48A elements (more complex than simple multipliers, they include post- and pre-adders and have cascading capabilities) (Xilinx, 2008).



(a) Spartan-3A XC3S700AFG484 (b) Spartan-3A DSP XC3SD3400AFG676

Figure 3.6: The two Xilinx boards with which the hardware simulator was implemented

A block diagram of the embedded system for the XC3S700A board, generated by the Xilinx Platform Studio (XPS), is presented in fig. 3.7. The system is almost identical for the the XC3SD3400A board, with the exception of the IP core for the second serial port (this board has only one serial port). At the centre of the system there is a Microblaze processor (Xilinx, 2009), whose initial data and instructions are stored in block RAM. Although the processor has the ability to run without an external RAM, the internal block RAM may not be sufficient to accommodate extensive programs. Moreover, as discussed previously, this type of RAM is also used to store the data for the neural network, leading to a competition for blocks of internal RAM between the processor and the network peripheral. In order to avoid these problems, the external 512 Mbit DDR2 SDRAM has also been included in the system, together with the interface module, so that this RAM may be loaded with the software that the processor needs to run, leaving the internal RAM to the neural network. As a result of the fact that XPS can program directly only the FLASH memory but not the external RAM, a FLASH module is also present. During the "boot" process, the contents of the FLASH memory are copied to the SDRAM and control is then transferred to the SDRAM. Of course, this bootloader (which is automatically generated by XPS) has to reside in block RAM. A small percentage of the internal RAM blocks must always be assigned to the processor for the instructions of the bootloader.

Besides the network peripheral (*net_peripheral_0*), there is also a module for the board's LEDs (*LEDs_8bit*), included here simply as a convenient way to signal possible anomalies during robot operation. Communication of the FPGA with the external world is achieved through the two serial ports (*RS232_DTE* and *RS232_DCE*). The DCE port has been configured at a data rate of 115200 bits/second and connects the board with a computer from which it receives

commands and to which it can send data (e.g. the current state of the network). The robot is connected to the FGPA via the DTE port, configured at 38400 bits/second (the robot's maximum data rate). The two remaining modules are a timer (*xps_timer*) and an interrupt controller (*xps_intc*), useful for measuring time, a functionality which the software cannot provide, since the Microblaze software is a simple, monolithic application, without the support of an operating system.

3.2.3 The graphical user interface

Manually creating a memory initialization file (MIF), like that of fig. 3.3, can be a very tedious task, except for the simplest cases. In order to automate this process, a simple tool was developed in MATLAB (fig. 3.8) which can load a network, described in an XML file, and subsequently create both the MIF files (one for each core) and the accompanying VHDL configuration files. XML offers the advantage of describing a network in a structured way so that a description is easy to read and possibly may be shared by different applications. It is an approach that has already been adopted by other research groups in the computational neuroscience community, e.g. NeuroML is an XML-based language which can describe neural networks at different levels of abstraction and detail in a standarized format (Gleeson et al., 2010).

As an example, a simple network with four neurons and two synapses, with only one neuron type and one synapse type, is described in XML in the following way :

```
<?xml version="1.0" encoding="utf-8"?>
<network>
   <configuration>
      <config_neurons>
         <neuron_type>
            <code>1</code>
            <model>izhi</model>
            <a>0.3</a>
            <b>-0.2</b>
            <c>-65</c>
            <d>8</d>
            <peak>30</peak>
            <rest_potential>-60</rest_potential>
            <rest_recov>-14</rest_recov>
            <thres_potential>-40</thres_potential>
            <cap>100</cap>
            <k>2</k>
         </neuron_type>
      </config_neurons>
      <config_syns>
         <synapse_type>
            <code>1</code>
            <gmax>7</gmax>
            <gmax_stdp>20</gmax_stdp>
            <vr>0</vr>
            <ts>2</ts>
            <pmax>1</pmax>
            <Aplus>0.2</Aplus>
            <Aminus>0.1</Aminus>
            <tplus>50</tplus>
```

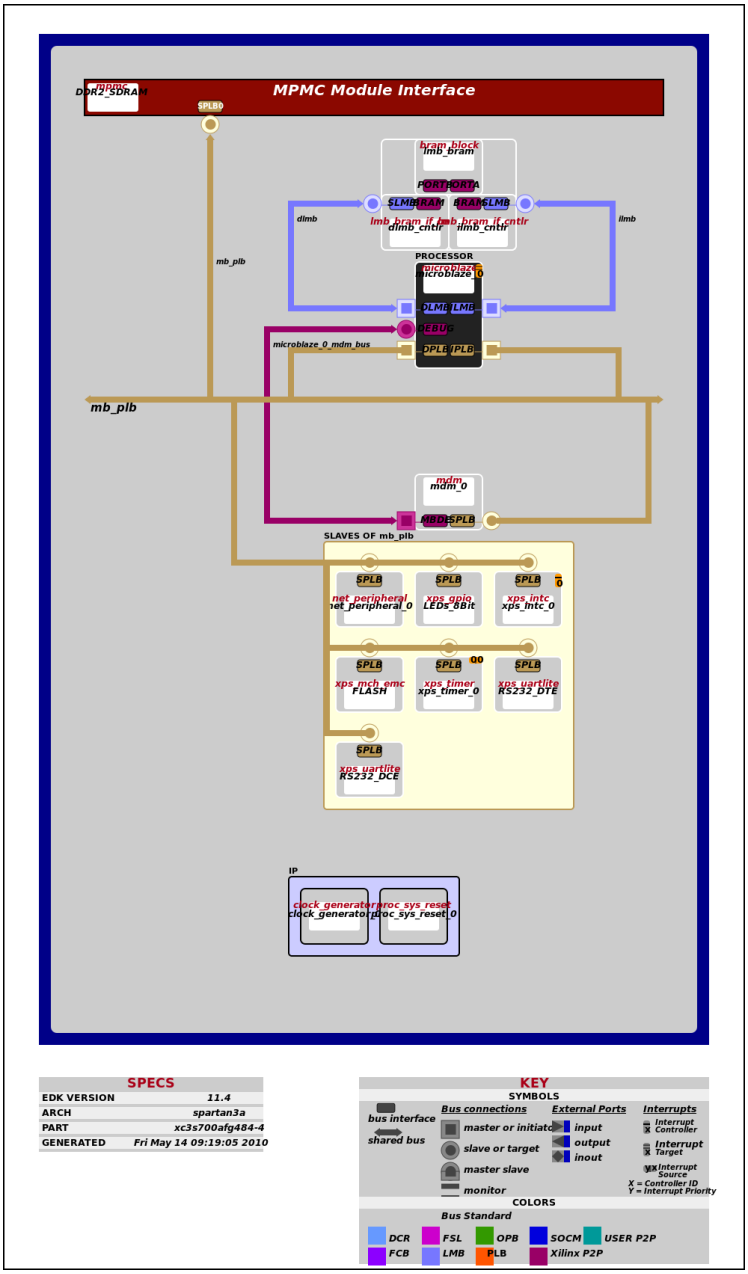


Figure 3.7: The embedded system which supports the neural network hardware simulator

```
            <tminus>5</tminus>
            <tc>10</tc>
            <vpost_peak>30</vpost_peak>
            <mode>1</mode>
         </synapse_type>
      </config_syns>
      <config_learning>
         <neuromod_type>
            <code>1</code>
            <tc>10</tc>
            <modc_max>1</modc_max>
         </neuromod_type>
      </config_learning>
   </configuration>
   <architecture>
      <layer id="1">
         <neuron id="1">
            <type>1</type>
            <input>true</input>
            <projection>
               <target>3</target>
               <type>1</type>
               <learning>-1</learning>
            </projection>
         </neuron>
         <neuron id="2">
            <type>1</type>
            <input>true</input>
            <projection>
               <target>4</target>
               <type>1</type>
               <learning>0</learning>
            </projection>
         </neuron>
      </layer>
      <layer id="2">
         <neuron id="3">
            <type>1</type>
            <input>true</input>
         </neuron>
         <neuron id="4">
            <type>1</type>
            <input>true</input>
         </neuron>
      </layer>
   </architecture>
```

```
</network>
```

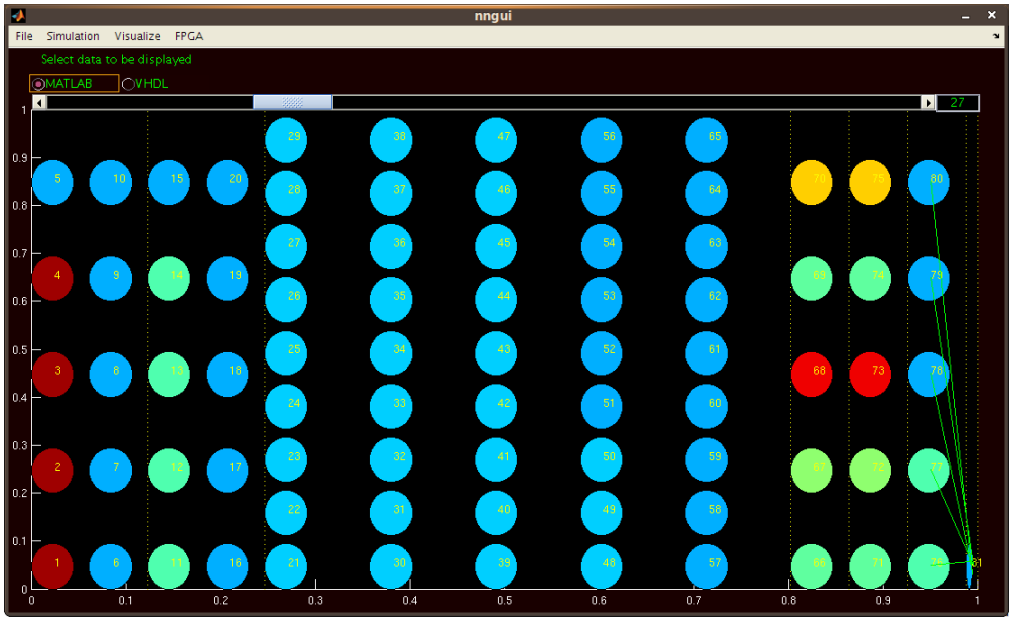


Figure 3.8: GUI of MATLAB tool for building and simulating neural networks, converting their XML descriptions to MIF and VHDL and communicating with the FPGA

When we need to implement a network with hundreds of neurons and thousands of synapses, it is still difficult to describe it, even in XML format. The tool includes a "wizard" which guides the user in a step-by-step procedure, so that the XML, MIF and VHDL files, even for large networks, may be produced in a matter of minutes, by making some relatively simple choices (number of layers, number of neurons per layer, connection probabilities etc). The only hardware related parameter that the user has to set is the number of cores for the FPGA simulator. After setting this parameter, the user also has the option to choose how many neurons should be assigned to each core. Since it may be difficult to estimate the computational load of each processor simply by the number of neurons which have been assigned to it, the user can let the tool determine the most efficient load distribution (see also section 3.3.3).

The tool also has the ability to simulate the behaviour of the network, albeit not very efficiently, so that the results from the FPGA can be checked for consistency against those from MATLAB. The results from the simulation can be stored for later processing but the activity of the network can also be viewed in a graphical manner (fig. 3.8). The user can load the results from the VHDL simulation or from the FPGA itself and switch between the two different views (software or hardware) for a more direct comparison.

Finally, the GUI can be used to control the KOALA robot (see chapter 4). Via the FPGA, the user can send motor commands to the robot and read the sensors as well. More important though is the ability to display and record the activity of the neural network (neurons' spikes)

while the robot is performing an experimental task. Therefore, with this GUI we have a more or less complete interface to the FPGA and the robot.

3.3 Testing the hardware simulator

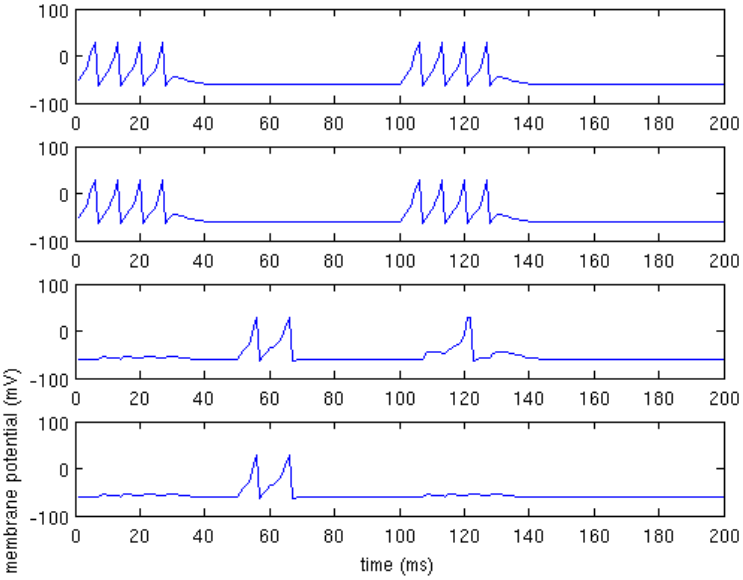
Before employing the embedded system described previously for robotic experiments, it was first tested in order to assess both its accuracy and its performance in terms of speed and hardware resources consumption.

3.3.1 Accuracy

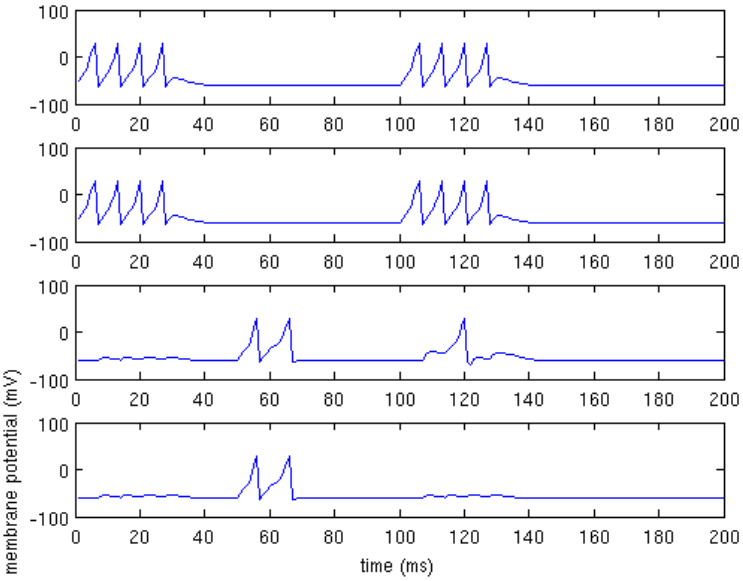
Of course, we cannot expect from the hardware design to be able to exactly reproduce the software simulation results. A MATLAB simulation can use high precision floating point arithmetic whereas the hardware version presented here represents values with 16 bits and performs fixed point calculations. When dealing with neural networks, timing may sometimes be a factor which greatly influences the waveforms of the membrane potential of neurons, e.g. a spike arriving a few milliseconds later may fail to generate another spike at the postsynaptic neuron. We may therefore see such discrepancies between the hardware and software versions, with some spikes missing or others appearing, but the overall spiking (and learning) patterns should remain similar.

For the simple network of four neurons and two synapses presented above, the results of the software and hardware simulations are shown in fig. 3.9 and 3.10. Initially, the first two neurons are triggered by an external current source for 30 milliseconds and shortly afterwards the second two neurons for 10 milliseconds. Beginning at the time point of 100 milliseconds, the first two neurons are again triggered for 30 milliseconds but this time no external stimulation of the second two neurons follows. The third neuron generates a spike because the conductance of the synapse projecting from the first to the third neuron has strengthened during the previous 100 milliseconds, following the STDP learning rule, whereas the other synapse has remained unaffected due to a lack of a learning mechanism (see *"learning"* node in XML description). As we can see in fig. 3.10, the results of the hardware version for the conductance are somewhat different than those of the software version. This is due to the timing of the third spike of the third neuron, which is followed by two spikes of the first neuron (hence the two decreasing steps) in the hardware case but by only one spike in the MATLAB version.

As an example of how neuromodulation works, we can consider again another network of 4 neurons with the same architecture but with STDP replaced by the learning mechanism of neuromodulation. Fig. 3.11 and 3.12 shows the behaviour of the network for 1200 ms, with neurons 1 and 2 being externally triggered from 1 to 100 ms and from 1001 to 1100 ms and neurons 3 and 4 from 401 to 500 ms. There's also a neuromodulator release "event" at the

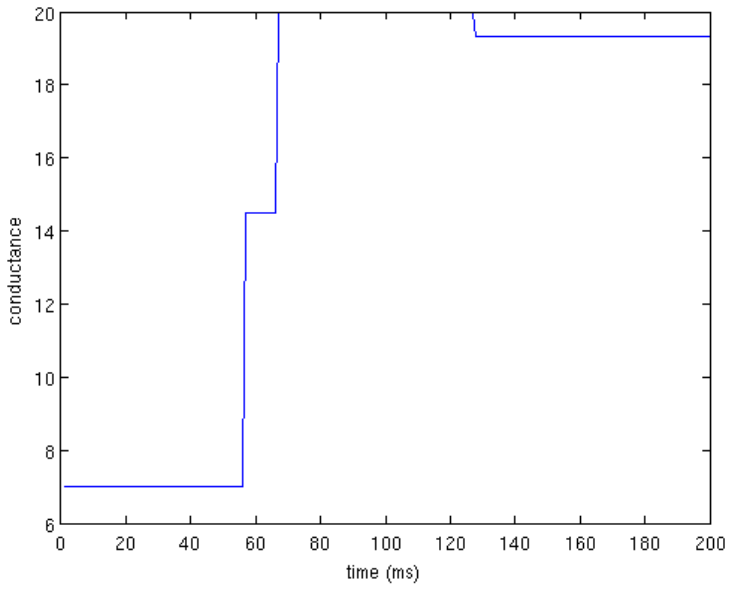


(a) Membrane potential of neurons 1 to 4, MATLAB version

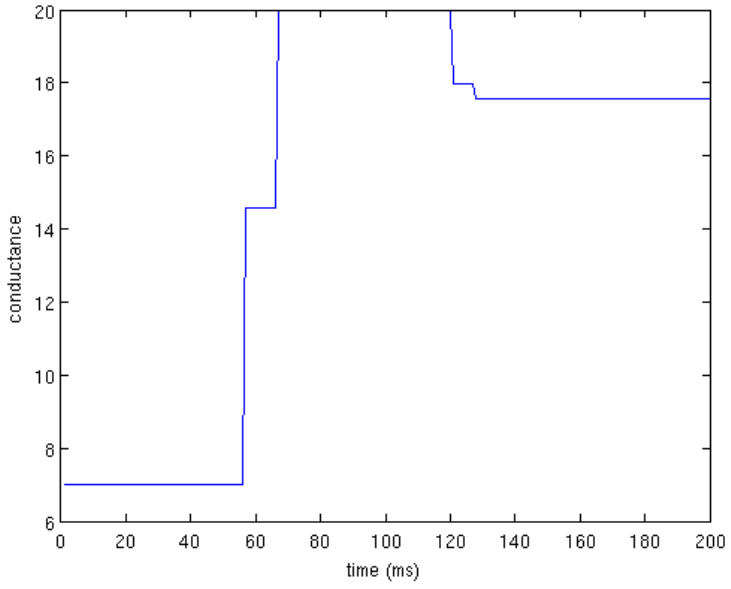


(b) Membrane potential of neurons 1 to 4, VHDL version

Figure 3.9: Comparison of MATLAB and VHDL versions of the simulator for a simple network of 4 neurons with respect to neuron membrane potentials



(a) Conductance of synapse from neuron 1 to neuron 3, MATLAB version



(b) Conductance of synapse from neuron 1 to neuron 3, VHDL version

Figure 3.10: Comparison of MATLAB and VHDL versions of the simulator for a simple network of 4 neurons with respect to synaptic conductances

time point of 801 ms. Figures 3.11(a) and 3.11(b) depict the activity of the neurons for the software and hardware versions whereas figures 3.12(a) and 3.12(b) show the neuromodulator concentration (left) and the conductance of the synapse from neuron 1 to neuron 3 (right).

3.3.2 Utilization of hardware resources

The percentage of the available hardware resources utilized by our design is not only an indicator of how efficiently it makes use of them but it also has a direct impact on the speed performance for the simple reason that the number of computational cores to be included is a user-modifiable parameter. Of course, the fewer slices a core requires, the more cores a board can implement, resulting in higher speedups (although this is not a rule without exceptions, see 3.3.3).

Another feature of the design is that the number of cores does not depend directly on the size of the network, unless there are specific requirements for the desired speedup. An increase in the size and/or complexity of the network results in more RAM blocks being employed but it does not affect the rest of the design, as depicted in fig. 3.13. For four networks of different size (25, 81, 217 and 306 neurons), we generated the bitstream for five different configurations (2, 4, 6, 8 and 10 cores). Only the XC3SD3400A FPGA was used, since the XC3S700A cannot accommodate more than two cores. It is obvious that the number of slices is more or less the same, regardless of network size and it is only affected by the number of cores. If we look more carefully, we can see that there is a slight tendency for the lines of smaller networks to lie below those of the larger ones but this is due to the fact that smaller networks usually have less neurons receiving external input, thus requiring fewer registers to hold the values of these inputs.

As far as the number of RAM blocks is concerned, it is clear that more blocks are required as the size of the network increases. The number of cores also seems to influence RAM utilization, though in different ways. For the networks with 217 and 306 neurons, a slight variation is observed as the number of cores changes but the number of RAM blocks does not deviate much from a “mean” value of 9 and 11 respectively. On the other hand, the smaller networks don’t have such a “mean” value but adding more cores tends to increase the number of RAM blocks they need. This behaviour can probably be attributed to the fact that many RAM blocks are “underutilized” in the case of small networks.

For example, if we assume that one RAM block has 1024 addresses, with a 16 bits width, and that we have a network distributed among 2 cores, each requiring 1000 addresses, then 2 RAM blocks are sufficient (fig. 3.14(a)). If we distribute the network among 4 cores, then we would have to use 4 RAM blocks (each core needs at least one RAM block), but they would only be half-full, each requiring 500 addresses (fig. 3.14(b)). If we now assume that we have a larger network which, when distributed among 2 cores, needs 4000 addresses for each core,

then it would occupy 8 RAM blocks (4 per core, fig. 3.15(a)). Distributing the network among 4 cores does not affect the number of required RAM blocks however. We would still need 8 of them (now 2 per core), all being almost full (fig. 3.15(b)).

3.3.3 Speed performance

Before discussing the results regarding the speed performance of the design, a few remarks about the way this performance is measured are necessary. It should first be noted that all values of the speedup are relative to real time and not to a software implementation, i.e. a simulation with a design whose speedup value is 1 should take as long as the "real brain" to complete (e.g. 100 milliseconds of neural activity would require 100 milliseconds of simulation time). The reason for this decision is that the efficiency of software implementations may vary significantly, depending on the machine they run on, the programming language used, the coding style of the programmer etc. In addition to that, the aim of the project is to attach these "silicon brains" to a robot and conduct behavioural experiments. Therefore, the main question is whether the design can exhibit real-time performance so that the robotic experiments can finish within a reasonable amount of time.

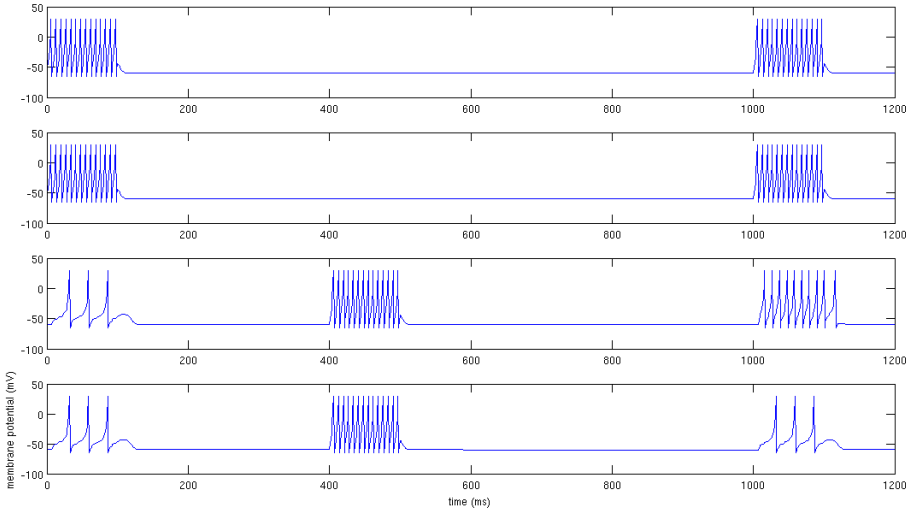
During a hardware simulation, the neural network attached to the Microblaze processor as a peripheral (see section 3.2.2) does not run continuously. Instead, the software driver first has to set the values for the external inputs of the network and then send an "update" command (pulse) to the peripheral. Subsequently, the peripheral updates the state of the network in the RAM blocks only for one simulation time step and control is then returned to the software application. If we want to run a simulation for 100 milliseconds, with a simulation time step of 1 ms, we have to repeat this loop 100 times, sending 100 "update" pulses. In order to evaluate the speedup of the design, simulations were run for a specific amount of time of 500 milliseconds (500 update cycles, with a time step of 1 ms) and the time needed for each update cycle to complete was measured, resulting in 500 such time measurements for each simulation run. The mean value of these 500 measurements was then used to compute the speedup.

As mentioned previously (section 3.2.2), the embedded system does not have an internal "sense of time". How are we then supposed to measure time? Using the timer peripheral is one way. Before sending an update pulse, the timer is started, it keeps counting while the software application is in the wait loop and when the update cycle finishes, its value is read and stored. This value is actually the number of clock cycles spent and not a time measurement but we can compute time simply by multiplying by the clock period. Another way to count the number of clock cycles is by including a hardware counter inside the network peripheral which starts counting as soon as an update pulse arrives. Although one might think that the results obtained from the two different ways of measuring speed performance should be identical, this is not the case, as fig. 3.16 shows.

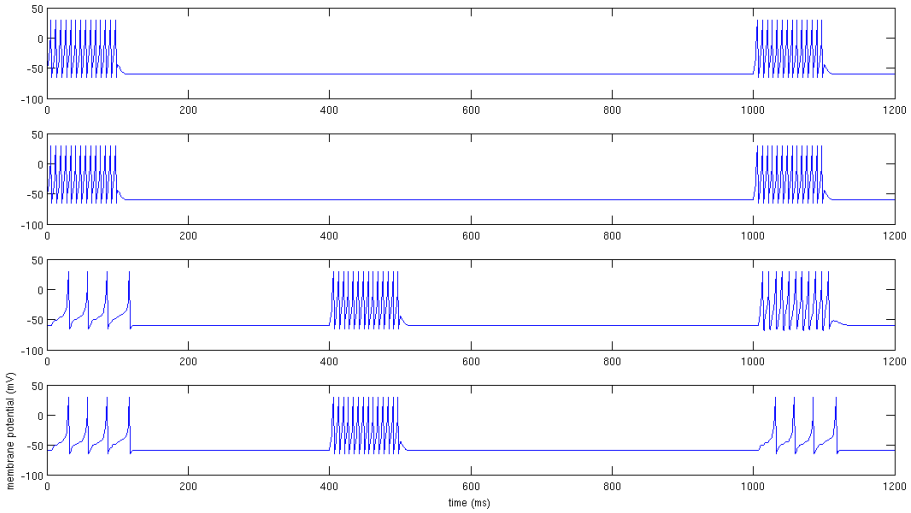
The speedup we get from the measurements of the software counter (software in the sense that its behaviour, like starting it, stopping it and reading its value is controlled by a software driver) are consistently lower than the ones of the internal hardware counter. In fact, the effect is even more pronounced in the case of smaller networks. The reason for this discrepancy lies in the delay that the software commands for the control of the external timer introduce. When dealing with timescales of microseconds, even just a few software commands can have a visible impact, especially when the system frequency is in the range of MHz. For the smaller networks which can be updated faster, the proportion of the total time consumed by the software delay (which is stable since the same software commands are executed) becomes even greater, thus leading to more significant deviations of the software from the hardware version. Although it would not be wrong to claim, based on the above analysis, that the results obtained from the internal hardware counter are more accurate, we should bear in mind that the network peripheral is always used, at least in the scope of this project, from within an embedded system and controlled by the software that runs on Microblaze. Therefore, not only is it impossible to ignore this software delay but the penalty that it imposes upon the whole system becomes even more important when we have to introduce more lines of software code between successive updates, as is the case when controlling a robot (see chapter 4).

It is also worth commenting the performance of the design for the network that has 25 neurons. Increasing the number of cores beyond 4 does not improve the speedup which remains the same. Until now, there has been no mention of how the computational load is distributed among the cores. Obviously, strong imbalances may appear, e.g. we may have one core being responsible just for one neuron and another core for twenty neurons and their input synapses. Such imbalances result in wasted cycles where some cores have to wait for the other, more heavily loaded cores to finish. In order to distribute the total load as evenly as possible among the cores, the MATLAB tool which creates the MIF files (see section 3.2.3) attempts to optimize this distribution by evaluating the computational cost of each neuron and assigning the neurons in such a way as to prevent any imbalances. The cost of a neuron does not depend only on the number of operations for updating the membrane potential and the recovery variable but on the cost of updating its input synapses as well (which is also different for synapses with no plasticity mechanism, for those with STDP and for those affected by a neuromodulator) because these synapses have to be assigned to the same core as their target neuron (section 3.2.1). This means that sometimes, when a network has neurons with many input synapses (high-cost neurons), then the load imbalances might not be able to be corrected beyond a certain point since the cost of a single neuron cannot be broken down. In such cases, adding more cores does not improve the overall performance because there is always one core who has been assigned a high-cost neuron and whose load remains the same. The performance of the other individual cores does improve (provided there is a margin for improvement) but they have to

wait for the overloaded core.

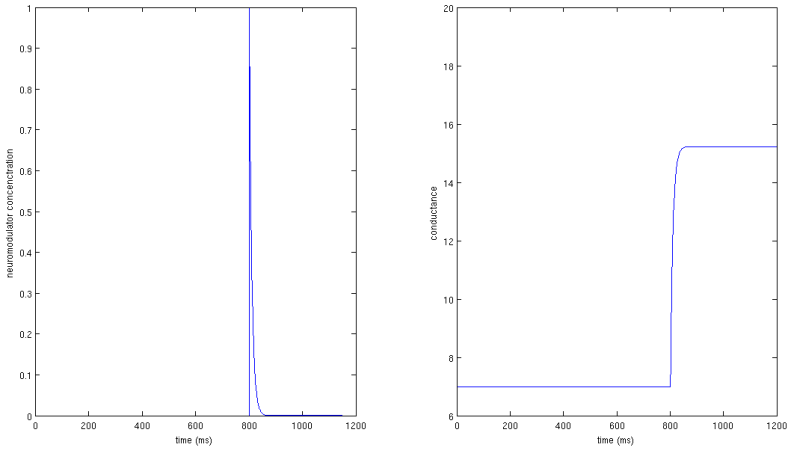


(a) Membrane potential of neurons 1 to 4, MATLAB version

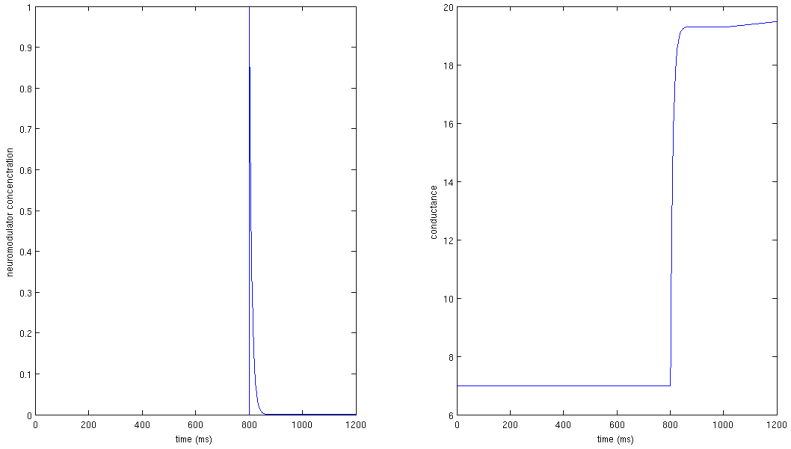


(b) Membrane potential of neurons 1 to 4, VHDL version

Figure 3.11: Comparison of MATLAB and VHDL versions of the simulator for a network of 4 neurons and one neuromodulator with respect to neuron membrane potentials

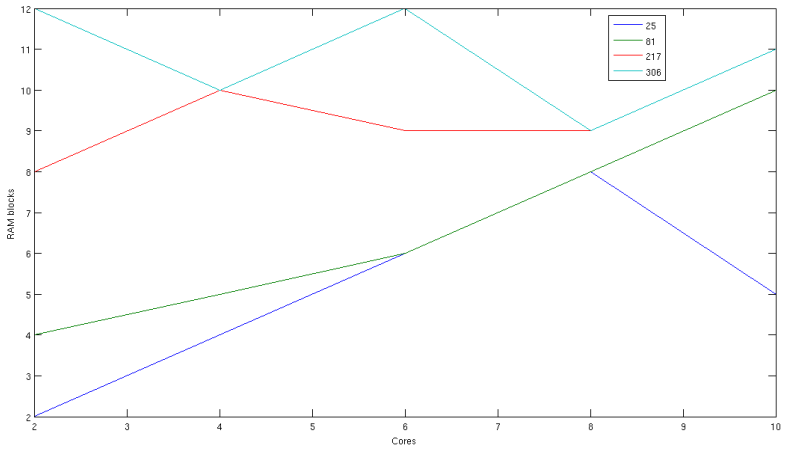


(a) Neuromodulator concentration and conductance of synapse from neuron 1 to neuron 3, MATLAB version

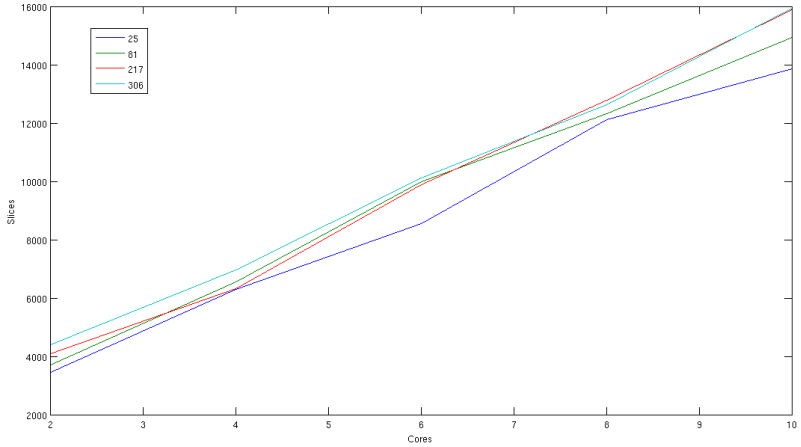


(b) Neuromodulator concentration and conductance of synapse from neuron 1 to neuron 3, VHDL version

Figure 3.12: Comparison of MATLAB and VHDL versions of the simulator for a network of 4 neurons and one neuromodulator with respect to neuromodulator concentrations and synaptic conductances

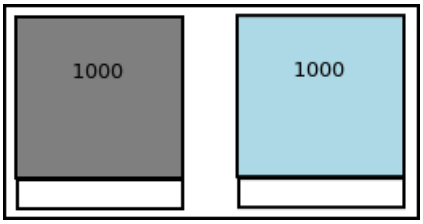


(a) Utilization of RAM blocks

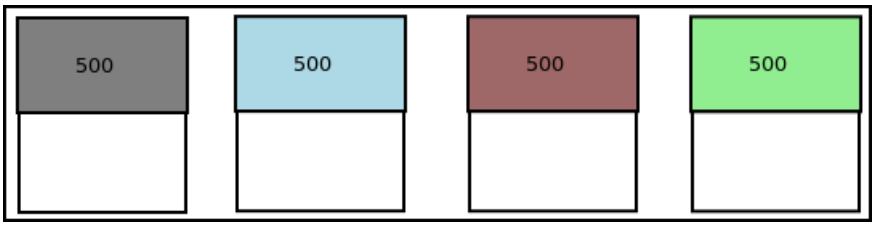


(b) Number of slices consumed

Figure 3.13: Hardware resources utilized by the simulator for 4 different neural networks consisting of 25 (blue line), 81 (green), 217 (red) and 306 (cyan) neurons as the number of computational cores is increased

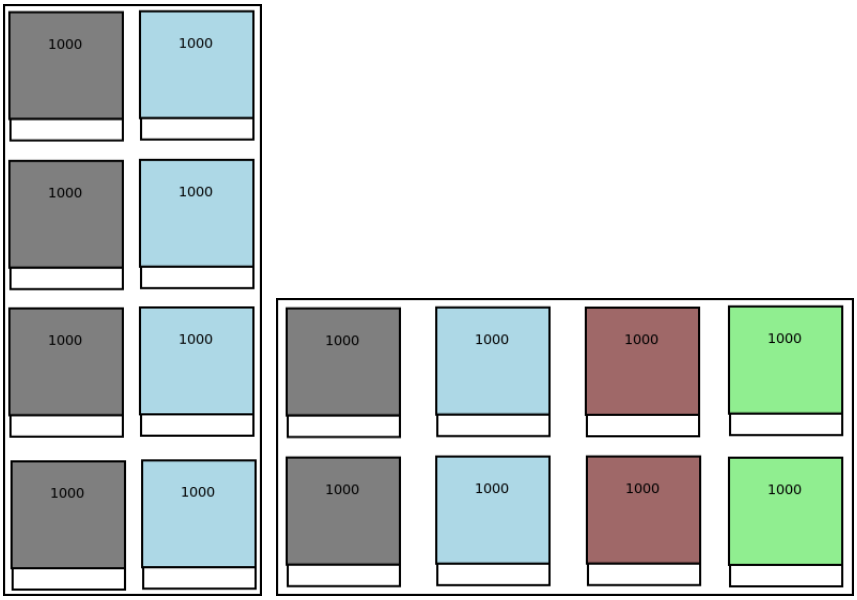


(a) Network distributed among 2 cores



(b) Network distributed among 4 cores

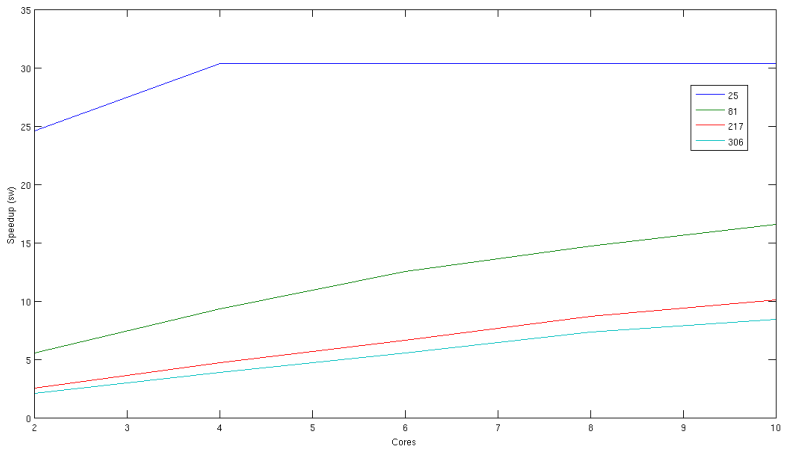
Figure 3.14: Possible occupation pattern of RAM blocks for a network requiring 2000 RAM addresses. Different colours indicate different cores.



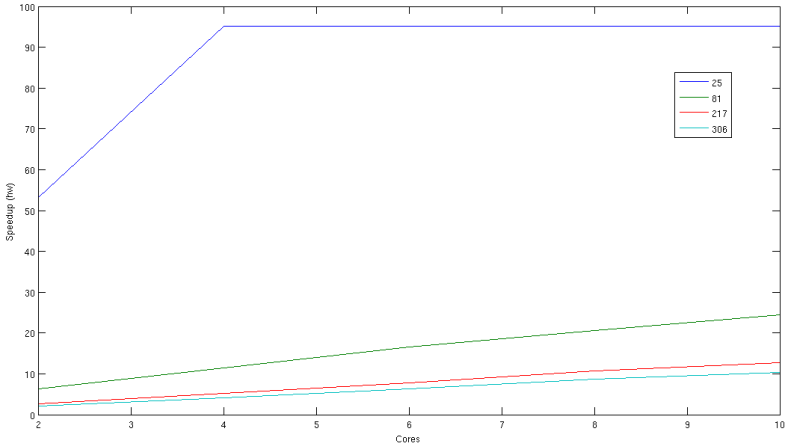
(a) Network distributed among 2 cores

(b) Network distributed among 4 cores

Figure 3.15: Possible occupation pattern of RAM blocks for a network requiring 8000 RAM addresses. Different colours indicate different cores.



(a) Speedup as measured by software counter



(b) Speedup as measured by internal hardware counter

Figure 3.16: Speed performance of the simulator for 4 different neural networks consisting of 25 (blue line), 81 (green), 217 (red) and 306 (cyan) neurons as the number of computational cores is increased

Chapter 4

Closing the loop

There is a certain tendency in the field of computational neuroscience to study neurons and their networks as if they were abstract, independent entities, residing in some outer space and stoically performing computations. This whole computer science jargon (information, computation, algorithm etc) might have become a second nature to many neuroscientists and treating the brain as a simple computational machine might seem as a self-evident methodological (if not ontological) rule. The traps to which this computational vulgarism can lead will not be discussed in this thesis (see (Canguilhem, 1968) and (Canguilhem, 1952) about a more general critique of the tendency to mechanize and physicalize biology and medicine, (Day, 2001) about a critical understanding of the concept of information [1], but also Zizek's remarks on the parallels between post-modern connectionism and post-fordist capitalism (Zizek, 2009)). Instead, we will focus our attention on another side of this approach that is problematic. The fact that the study of neural networks in a context-free environment is able to provide us with insights up to a certain point beyond which we can only get diminishing rewards.

To a post-aristotelean, "enlightened" mind, with its patronizing attitude towards Aristotle's natural philosophy, teleological explanations surely appear as unscientific. This same mind could also claim that it is just a matter of convenience when medicine and biology use the term "purpose" to describe the functions of cells, tissues etc and nothing teleological is implied.

[1]

> *...the historical construction of information (as fact, as re-presentation) erases freedom in historicity, or, that is, determines freedom to be agency within a set of known or knowable choices. (E.g., information theory's "freedom of choice"). Such a sense of knowledge extends to history itself, so that history is only possibility, not potentiality. Radical alterity and thus a radical sense of freedom (promised in the Enlightenment) is foreclosed.*

By extending this logic to neuroscience, we could argue that the interpretation of the brain as an information processing machine is just a disguised form of behaviourism that cements and consolidates the outer "order of the world" within a living organism, i.e. a radical internalization of that which is as it is, a new and more powerful form of control (although not so new, (Marcuse, 1964)). Learning becomes almost synonymous to control, (Holzkamp, 1995).

Again, we will not enter into an argument about the philosophy of medicine and biology and certainly not attempt to discuss epistemological questions about the meaning of (convenient) explanation. However, we have reasons to believe that the activity of a neural network can be better understood when the "purpose" (with or without quotes) of its functioning and the behaviour of the organism whose part it is are taken explicitly into account, provided that we do not succumb to some kind of obsolete behaviourism.

Of course, this thesis is not so ambitious as to claim that the work presented in it accomplishes such a task. Just a few preliminary and clumsy steps towards this goal comprise the content of this chapter. In fact, it mostly raises some new questions rather than giving answers to the old ones. Sometimes, reformulating old and posing new questions is already a first step towards an (always temporary and constantly under revision) answer.

4.1 Biorobotic platform

If we are to follow this path, the question that naturally follows is how we take behaviour into account when modelling neural networks. Building a software model of both a behaving agent together with its "brain" and its environment is one way (Beer, 2008). One problem with this approach is that we are usually restricted to simple "worlds" whose modelling is based on many assumptions and which can be easily manipulated to fit our expectations. There is no doubt that we can build more complex and realistic worlds, with some considerable effort. Alternatively, we can simply go directly for the "real" thing, i.e. implement a physical robotic system with a silicon "brain".

For this purpose, we have chosen to use the KOALA robot (fig. 4.1(a), *www.k-team.com*). The aim of this project is to study the neural structures that are involved in learning and not to develop models of the fly's motor (and certainly not flying) circuits. Therefore, a relatively simple robot, such as the KOALA, is more than sufficient for our purposes. It is a mid-size robot that can host an FPGA board while having a simple communication interface and motor capabilities, allowing us to focus on the FPGA without having to deal with potentially complex issues of motor control.

4.1.1 Experimental setup

The kind of behaviour that we would like our robot to be able to reproduce is necessarily determined by the kind of behavioural experiments which test learning and memory in insects (flies in our case, see 2.3). One of the most widely used experimental setups (by our group as well) is the T-maze. Suitable mostly for conditioning experiments, its aim is to make a fly associate certain odours (or mixtures) with an electric shock. Simple as it may seem, such a setup is a good starting point for running biorobotic experiments, exactly because of its

simplicity.

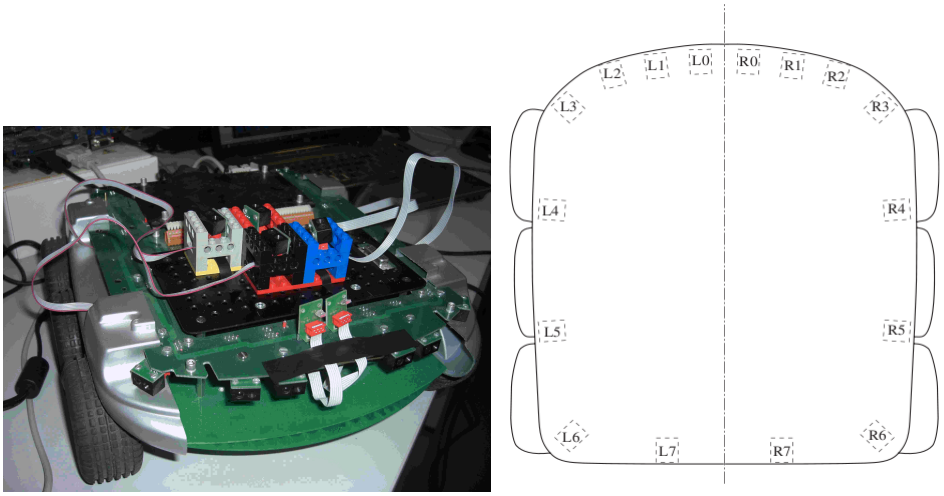
Fig. 4.2(a) shows the T-maze in which flies are trained in experiments for various conditioning paradigms while fig. 4.2(b) shows the arena which we built for our robot experiments. Of course, there's no point in trying to replicate the T-maze in its exact form but we tried to keep the same overall shape. As in the case of flies in the T-maze, the robot in the arena has two choices. It can either go to the right or to the left "testing tube". Contrary to the T-maze though, where training takes place before testing in a separate tube, the robot has to establish associations while "exploring" the arena. Flies are placed in the training tube, they receive a sequence of electric shocks in the presence of an odour and they are subsequently forced to make a choice among the testing tubes. This experimental protocol was considered too static and trivial for the KOALA which was let free to wander around the arena.

The aim of an experiment in the arena is to assess whether the robot can learn to associate certain stimuli with the presence of a light source and how easy it is to discriminate between different stimuli. A light source inside a tube (US) elicits an escape response which could be interpreted as the unconditioned response (UR). Success is therefore defined as the acquisition of the ability to produce a conditioned response (CR), i.e., an escape response even when the light is turned off.

4.1.2 Sensors

As far as sensory inputs are concerned, the KOALA does not offer any olfactory sensors with which to perceive odours. In fact, an artificial nose is probably one of the hardest sensors to build, at least in the form of a cheap, readily available component. Therefore, we have to substitute olfaction for another "modality" while still treating the sensory inputs as odours, i.e. the mechanism under investigation and the corresponding neural models are still those of olfaction and olfactory learning. For the experiments presented in this thesis, the infra-red (IR) proximity and ambient light sensors were used in order to detect the presence of obstacles and stimuli. Sixteen sensors are positioned around the KOALA and can measure either the light reflected by obstacles or the ambient light (K-Team, 1999) (fig. 4.1(b)).

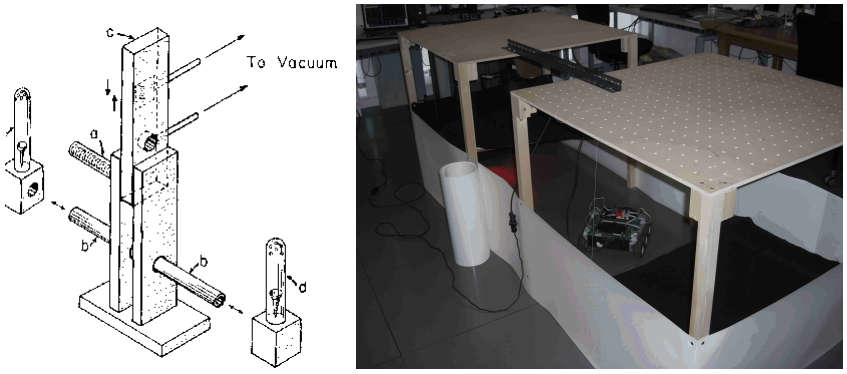
During operation, the robot needs to be able to distinguish between three different kinds of stimuli. First of all, some basic obstacle avoidance functionality is required so that the robot may keep away from the walls of the arena. This is achieved by taking IR measurements from six of the front sensors (L1, L2, L3, R1, R2, R3). The remaining two front sensors (L0, R0) operate only in the ambient light mode, functioning as detectors for any light sources. A source of light in the environment, as sensed by these two sensors, acts as a punishment signal (unconditioned stimulus, US), resulting in a neuromodulator release. "Odors" constitute the third type of stimulus (conditioned, CS). In order to use the IR sensors as "odor" detectors without hindering the movements of the robot inside the arena at the same time, the four side



(a) The KOALA robot, modified for the purposes of the project

(b) Position of the IR and ambient light sensors around the KOALA (K-Team, 1999)

Figure 4.1: The KOALA robot



(a) The T-maze, as designed by (Tully and Quinn, 1985)

(b) The arena for the robot experiments

Figure 4.2: The T-maze and the arena

sensors (L4, L5, R4, R5) were removed from their initial positions, reconnected with longer cables and placed on top of the KOALA, pointing upwards (fig. 4.1(a)). This repositioning provides us with the ability to detect objects lying above the KOALA as well as their distance from the floor (if they are sufficiently close to the sensors). Objects at different heights may then be interpreted as different odours and an odour can also be combined with a punishment, i.e., the odour and the punishment can be simultaneously sensed by the robot.

4.1.3 Interface to FPGA

The robot is controlled by the embedded system described in 3.2.2 through a serial connection. Due to time constraints, only the board with the XC3S700A FPGA was employed since it is equipped with two serial ports, rendering the establishment of communication links with both the robot and the GUI MATLAB tool much easier. A connection between MATLAB and the other board (XC3SD3400A) would require the instantiation of an Ethernet controller and the use of the xilkernel operating system. The downside of running the neural network on the XC3S700A FPGA is that only two computational cores can fit in the design (a substantial percentage of the slices are consumed by the Microblaze processor and the controllers for the various peripherals) and that we are limited to small networks because of the few RAM blocks available (11 from a total of 20 RAM blocks are actually reserved again by the Microblaze). For more advanced experiments however, it is imperative that the XC3SD3400A or an even more powerful FPGA be used.

The second serial port connects the FPGA with the GUI tool so that we can watch and record the activity of the neural network while running an experiment. Currently, the GUI is capable of recording only the activity of the neurons or more precisely the spikes of neurons (not the exact value of the membrane potential). Although the Microblaze software application can read any memory address of the RAM blocks, it was decided that only information about the spikes (their position in time) should be sent back to the GUI due to the low data transfer rate of the serial connection. Assuming a rate of 115200 bits/second, if we wanted to record the activity of ten neurons by transmitting the exact values of their membrane potential, then we would have to send 160000 bits for 1 second of neural activity (16 bits for each value, 1000 values for each neuron in 1 second, assuming a time step of 1 millisecond, 10 neurons in total). This means that the transmission would need almost 1.4 seconds just for 1 second of neural activity and for only 10 neurons. On the other hand, even if we assume a spike frequency of 20 Hz for all of the 10 neurons, then sending only the spikes' position would require less than 28 milliseconds (one 16 bit value is sent for each of the 20 spike positions and for each neuron). A future design with an Ethernet connection should be able to transmit more data about the state of the network, including for example synaptic conductances or the neuromodulator concentration.

4.1.4 The robot's "brain"

Ideally, we would like to be able to simulate the whole olfactory pathway of the fly's brain with all its different structures and connections. Even if the available hardware was powerful enough to host a neural network of this size and complexity, we would still have to make many assumptions since many of the involved structures, with the exception of the antennal lobes and the mushroom body, have not been thoroughly studied. Considering the limited resources offered by our FPGA board, the ambitions of this thesis are much more modest. Despite the small size of our robot's brain, we did try to build a neural network based on the fly's brain, including most of the structures which are part of the olfactory pathway.

In contrast to previous models, like those presented in sections 2.4.1 and 2.4.2, we are not going to be concerned with low-level details of neuron morphology and precise timing. We are mostly interested in how the overall architecture of the olfactory system affects its computational properties with regard to learning and memory. Therefore, the model presented below has more similarities with the ones discussed in section 2.4.3. All of these models are based more or less on the same neural architecture according to which the PNs constitute the first layer, diverging onto significantly more numerous KCs and then converging again onto the "output" layer of the ENs. At the same time, another parallel path passes through the LH. Our model complies with this general rule but with the important addition of an extra layer, located in the LH as well, which connects through a "loop" the PNs and the ENs (see below for details). Moreover, the purpose of our model is to be able to function as a "brain" of a robotic agent, with specific behavioural requirements. This means that certain, more abstract properties of the olfactory system, like its classification capabilities, investigated in previous studies, are less important. For our purposes, it is necessary to have a value signal, having as its main function the association of various stimuli, as in (Smith et al., 2008). However, the learning mechanism that was chosen (STDP with neuromodulation) was different from those found in previous studies due to the need to solve the so called distal reward problem. Finally, it should also be noted that our model differs from those of section 2.4.3 in another point. Learning occurs through synaptic weakening rather than strengthening, as explained below.

Our tiny fly brain consists of just 81 neurons, arranged in 5 layers (fig. 4.3, also fig. 3.8). The connections displayed in fig. 4.3 constitute the fundamental functional path which an input signal traverses from the first layer to the CR neuron. The path begins at the stage of the projection neurons (PNs, neurons 1 to 10) but future models should also include the stage of the olfactory sensory neurons (OSNs). As we have seen (2.3.5), the antennal lobes cannot be conceived as a simple relay station but play a more active role in shaping the stimulus signal. For the moment and until more neurobiological data become available that could drive a modelling attempt, we restrict ourselves to PNs.

Projection neurons from the antennal lobes target Kenyon cells (KCs, neurons 21 to 65)

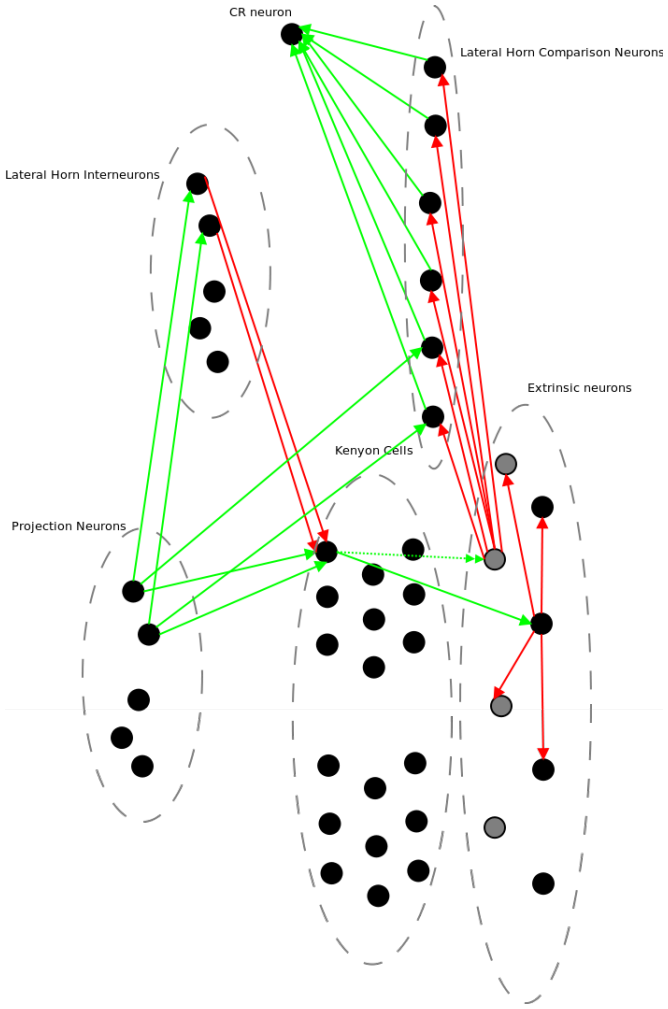


Figure 4.3: Architecture of the neural network, used as the robot brain. Green lines correspond to excitatory synapses whereas inhibitory connections are indicated by red lines. A dotted line means that the synaptic strength of this connection can be modified. The picture shows only one path for the neural signal. The same path is repeated for every combination of two PNs (see text for details).

in the mushroom bodies (MBs) both through direct connections and indirectly via the lateral horn interneurons (LHIs, neurons 11 to 20) (see section 2.2.3). KCs are thought to act as coincidence detectors and as a consequence their firing patterns are sparse and rare (fig. 2.3in section 2.2.4). In order for the KCs of our network to exhibit this behaviour, the time constant of the neurotransmitter decay for the KC to PN synapses was set to a very low value (2 ms) so that the effect of a single PN spike vanishes quickly. Additionally, PN activity generates an inhibitory drive, coming from the LHIs, which acts as a reset mechanism for KCs.

Some parameter tuning (with the conductances and neurotransmitter decay time constants of the PN to LHI synapses) was necessary in order to prevent the LHIs from firing prematurely and to delay their spikes until after the targeted KC has fired first. This behaviour was achieved by setting the PN-LHI time constant to a significantly higher value than that of the PN-KC synapses while lowering at the same time the value of the conductance (see table 4.1) so that the LHI neurons require more time to build up the necessary potential. On the other hand, the PN-KC and LHI-KC values are comparable because we want the LHI-KC inhibitory drive to act as a resetting mechanism, having a similar but delayed and "inverted" effect with respect to the excitatory PN-KC connections. The effect of these mechanisms is that the initial PN activity becomes more sparse and rare upon reaching the MB. Fig. 4.4 shows this effect. While 40% of the PNs are active, only about 13% of the KCs respond (fig. 4.4(a)). Moreover, KCs fire only once although PNs fire 4 times (fig. 4.4(b)).

Plasticity is a feature only of the synapses that project from the MB to the extrinsic neurons (ENs, neurons 66 to 75). The existence of STDP among synapses that project from KCs to ENs has been experimentally confirmed in the locust brain (Cassenaer and Laurent, 2007) (see also discussion in section 3.1.2). The ENs are divided into to two groups with different functionalities. One of them is the group of ENs (EN1) which project to the lateral horn comparison neurons (LHCNs, neurons 76 to 80, grey circles in fig. 4.3), the last layer of the network. These are the only neurons whose input synapses are plastic (dotted line in fig. 4.3). The role of the second group (EN2) is to produce some intralayer suppressing drive by sending inhibitory synapses to their neighbours in order to prevent widespread activation of the ENs layer and make the winning patterns more distinct. Therefore, activity in the PNs results in the emergence of a winning pattern among the ENs.

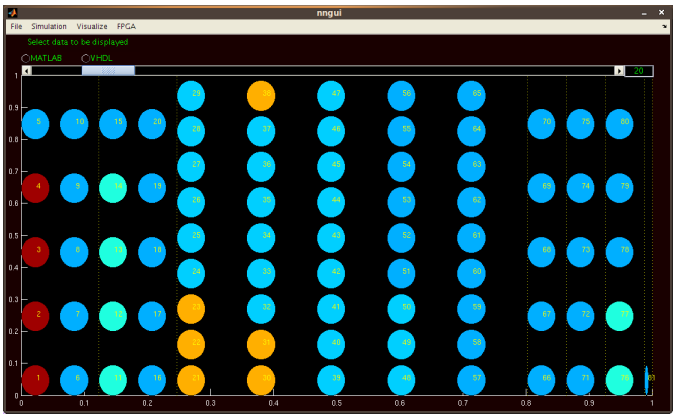
Due to the fact that the EN neurons have a high fan-in (from several KCs), it was necessary for the KC-EN synaptic conductances to be set to low values in order to avoid overexcitation of the ENs. For this reason, they were randomized within the range of 3-10 nS. The desired timing for the learning process was achieved by using a relatively long STDP time window (50 ms) and by setting the value of the synaptic tag decay time constant in the scale of seconds (1500 ms), as shown in table 4.1. With these values, the time difference between the CS and the US can be up to a few seconds and a significant learning score may be observed usually

Parameter	PN-KC	PN-LHI	LHI-KC	KC-EN1	PN-LHCN	EN1-LHCN
g_{max} (nS)	10	6	10	3-10	3	20
g_{max}^{stdp} (nS)	30	30	30	30	30	30
v_{rev} (mV)	0	0	-90	0	0	-90
t_s (ms)	2	20	3	20	20	40
p_{max}	1	1	1	1	1	1
A_+	0.2	0.2	0.2	0	0.2	0.2
A_-	0.1	0.1	0.1	0.2	0.1	0.1
τ_+ (ms)	50	50	50	1	50	50
τ_- (ms)	5	5	5	50	5	5
t_{ctag} (ms)	10	10	10	1500	10	10

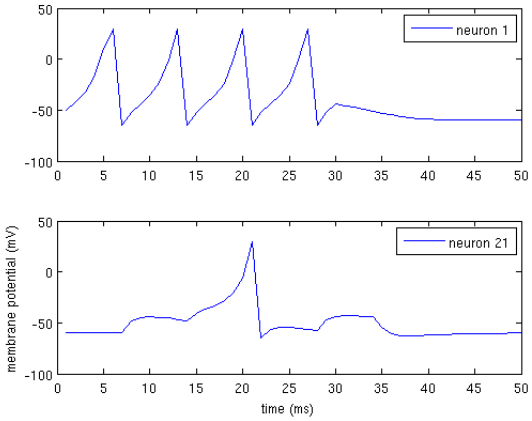
Table 4.1: Values of synapse parameters of the robot brain

within less than 5 trials. It has to be noted that, since only the KC-EN1 synapses are plastic, the values for the plasticity related parameters in tables 4.1 and 4.2 take effect only for these synapses. The parameter values for the other synapses are included in tables 4.1 and 4.2 for the sake of completeness because they may also be found in the XML file.

The role of the network's last layer (LHCNs) is to somehow compare the activity in the PNs layer with that in the ENs. LHCNs receive input both from the PNs (excitatory) and the ENs (inhibitory). Prior to learning, inhibition from the ENs, as a result of a stimulus in the PNs, manages to cancel out the excitation from the PNs. The result of learning is to make the synapses from the MB to the ENs weaker (the synapses' strength can only decrease) which in turn decreases the activity of the winning ENs pattern or even make it disappear completely. This quieting of the ENs dis-inhibits the LHCNs which can then excite the CR neuron, signalling the importance of the (now learned) stimulus. As shown in table 4.1, the inhibition of the EN1-LHCN connections is both stronger and more long-lasting than the PN-LHCN excitation so that the CR neuron may start firing only when learning has been truly substantial. In order to ensure that synapses can only become weaker, the strengthening portion of the STDP window (fig. 3.1) has been "removed", by setting the A_+ parameter to 0 (eq. 3.7). There has been no mention in the literature review of the existence of such neurons in the LH or of a plasticity mechanism which only decreases synaptic strength. However, some recent but yet unpublished data support the hypothesis about the existence of a special class of LH comparison neurons with inputs from both the PNs and the ENs and with a learning mechanism which is based on synaptic weakening rather than strengthening (personal communication with Stijn Cassenaer). Tables 4.1 to 4.4 present the values for the various parameters of the network.



(a) Sparsening effect in the MB. Different layers are separated by yellow, dotted, vertical lines. Whereas 40% of the PNs are active (1st layer), only 13% of the KCs respond (3rd layer).



(b) Neural activity among the KCs (bottom figure, neuron 21) becomes more rare than among PNs (top figure, neuron 1)

Figure 4.4: Transformation of neural activity from PNs to KCs

Parameter	**LHCN-CR**	**EN2-EN2**	**KC-EN2**
g_{max} (nS)	15	30	3-10
g_{max}^{stdp} (nS)	30	30	30
v_{rev} (mV)	0	-90	0
t_s (ms)	5	30	20
p_{max}	1	1	1
A_+	0.2	0.2	0
A_-	0.1	0.1	0
τ_+ (ms)	50	50	1
τ_- (ms)	5	5	1
t_{ctag} (ms)	10	10	1

Table 4.2: Values of synapse parameters of the robot brain, cont.

a	0.3
b	-0.2
c	-65
d	8
v_{peak} (mV)	30
v_r (mV)	-60
v_t (mV)	-40
C	100
k	2

Table 4.3: Values of neuron parameters of the robot brain

c_{mod}	1
t_{mod} (ms)	10

Table 4.4: Values of neuromodulator parameters of the robot brain

4.1.5 Control algorithm

It has already been mentioned that the robot's "brain" cannot assume full control of the robot but is responsible only for establishing associations between stimuli. The necessary pre- and post-processing (feeding the neural network with sensory input, reading its output, making the motor decisions) are actually performed by a simple control algorithm, written in C and running on the Microblaze processor. Fig. 4.5 shows a block diagram of the robotic system along with the information flows controlled by the algorithm.

This algorithm is composed of three loops, each with a different time period. The innermost loop is the simplest, its main role being the update of the network's state for one simulation time step (1 ms). It first checks for the presence of a stimulus (CS, US or neutral stimulus) and then sets the network's input, according to which stimuli are present. For example, a CS could result in the first four input neurons being triggered or the US producing a neuromodulator release. After setting the input, an update pulse is sent to the network peripheral and upon completion of the update cycle the state of those neurons that have been "tagged" to be of interest is recorded, i.e. the position of the spike (if there is one) within the recording time window (see below, also section 4.1.3) is stored to an array.

The next loop is the one which actually decides whether there is a stimulus in the environment or not and also determines the next motor command. It has a time period of 100 ms and at the start of each repetition it sends a command to the KOALA in order to read its IR and ambient light sensors. The IR values of the sensors L4, L5, R4 and R5 (see section 4.1.2) correspond to certain stimuli, e.g. in the context of elemental learning (A+ B-), a mean value between 150 and 300 may be interpreted as stimulus A being present whereas a mean value between 600 and 900 as stimulus B. This way, objects at different heights can be corresponded to different stimuli-"odours". It has to be noted that combinations of odours (like AB) are not directly mapped to combinations of environmental cues since we cannot combine objects at different heights. Instead, whenever we need such combinations, we use a single object which is simply interpreted as a compound stimulus. Likewise, the ambient light readings of the two front sensors L0 and R0 provide us with information about the US. If the readings are above some threshold, then it can be inferred that a light source is in the same area and the robot is "punished" (US present, neuromodulator release).

The rest of the sensors (L1,L2,L3,R1,R2,R3) are used for obstacle avoidance. The IR readings from these sensors can tell us whether an obstacle blocks the way and the direction that the robot should follow in order to avoid it. However, the decision about the new direction is not based only on the current values of the IR sensors but on the weighted sum of the last 15 values. Incorporating a small sensory "memory" like this helps us to deal with fluctuations in the environment which could mislead the robot into a strange and repetitive behaviour (e.g. in corners). According to these motor rules, the robot should move forward until it meets

an obstacle and turns to avoid it. There are two exceptions to this. First, if the robot has been moving forward for 100 consecutive repetitions of the loop (10 seconds), then it takes a random turn. Second, in case a US is present or the CR neuron has fired, an escape response is initiated (no distinction is made between the UR and the CR). The robot turns quickly 180 degrees and starts moving away and at the same time stops accepting new input for a certain period of time.

Finally, after reading the sensors and sending the appropriate motor commands, the loop waits until the previous, innermost loop has repeated 100 times which corresponds to 100 network updates or 100 ms of neural activity. This kind of synchronization between the two loops results in real-time robot behaviour by making 100 ms of behaviour correspond to 100 ms of neural activity. Of course, we could have merged the two loops by repeating this pre- and post-processing not every 100 ms but every 1 ms, with each update cycle. Once again, the limitations of the serial communication do not allow for something like this since the delay that it introduces is unacceptably high. This is the reason why sensor and motor commands are sent only once every 100 ms, a time period which is long enough to fit both the required software instructions and the corresponding 100 network updates but short enough so that the robot does not miss any important sensory elements.

The outermost loop has the longest period which is 1000 ms. Its function is simply to transmit the recorded data (neuron spikes and sensor values) back to the GUI. Before the transmission begins, the KOALA is shutdown (it stops moving and the two loops described above come to a halt) and it resumes its operation after the end of the transmission. This temporary shutdown was deemed necessary because the transmission may last from somewhere between hundreds of milliseconds to a couple of seconds when the GUI has to empty its buffers and write the data to temporary files.

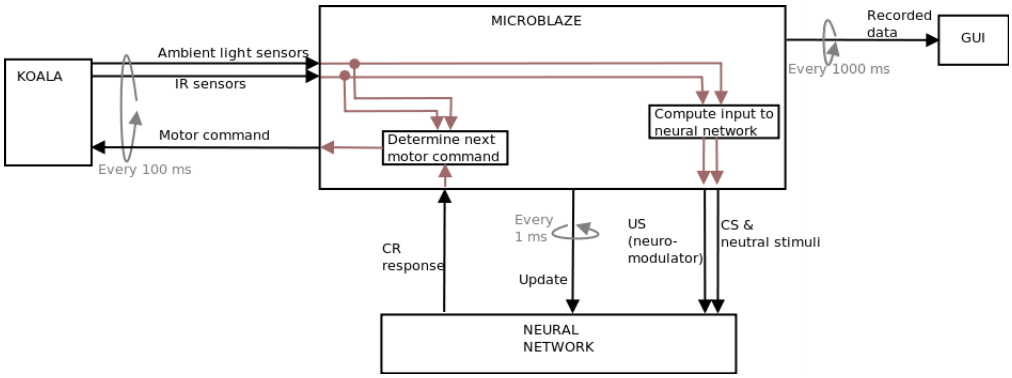


Figure 4.5: Block diagram of the robotic system

4.2 Experiments and results

A series of experiments was run with the robotic platform presented above in order to assess its performance in solving certain learning tasks. The number of experiments was relatively limited. The reason for the decision not to run many additional experiments was that the current experimental setup has certain limitations that prevent us from investigating certain interesting issues that would be worth examining (see the discussion in section 4.3.1). These limitations are mostly of a technical nature and should therefore be "easy" to overcome in future versions of the platform (section 5.1).

4.2.1 Learning capabilities of the robot brain

Despite its small size, the brain of our robot has some interesting learning capabilities, under certain restrictions. The results from testing the network for different learning paradigms are shown in fig. 4.6 - 4.11 [2]. The paradigms are elemental learning (A+ B-), mixture learning (AB+ CD-), discrimination learning (AB+ BC-), positive patterning (AB+ A- B-), negative patterning (A+ B+ AB-), biconditional discrimination (AB+ CD+ AC- BD-) and blocking (A+ AB+) [3]. These paradigms were chosen because they have also been used in behavioural experiments in order to test the learning capabilities of an insect's brain and thus provide us with some clues about the applicability of certain, more abstract models. For example, a simple elemental theory of learning (see section 2.1) could not account for success in a negative patterning test since reinforcing both A and B would necessarily increase the associative strength of AB.

In order to analyse how the network behaves under these paradigms, we focused our attention on the ENs. Although it is the CR neuron that is behaviourally relevant, it is not suitable as an indicator of the gradual effect that learning has on the network. It mostly functions in a "binary" mode, being quiescent for neutral stimuli and firing at its maximal frequency for learned input patterns. For this reason, the information we can get from the ENs is more useful. As we have already mentioned, a specific pattern emerges among the ENs as a result of presenting the network with a neutral stimulus and training has the effect of decreasing this activity. Therefore, the percentage of this decrease can be interpreted as a measure of robustness whereby the best performance is achieved when the original, "naive" ENs' pattern disappears completely after training. The plots presented in fig. 4.6 - 4.11 show how the response of the ENs to the various stimuli evolves during successive training trials. The score at each data point is computed as the decrease percentage of the ENs activity, using the naive response as

[2] The network was simulated on the XC3S700A FPGA.

[3] A, B, C and D are different "odours" - input patterns. The plus symbol denotes reinforcement. The network is considered to be successful in accomplishing a task if it responds only to the reinforced signals after some training trials.

the base value. A score of 100% for a specific input pattern means that it does not elicit any response at all after training.

In the case of elemental learning, we can observe that the reinforced stimulus achieves a perfect learning score whereas the response to the neutral stimulus remains unaffected. In the discrimination learning paradigm, stimulus AB follows the same pattern as A in the elemental case, but the neutral stimulus (BC) is also slightly affected, as a result of its overlap (50%) with AB. This interference between the reinforced and the neutral stimuli becomes more pronounced in the paradigm of positive patterning. The compound stimulus AB, which the network is expected to learn, does indeed achieve a score of 100%. At the same time, however, the individual element A becomes indistinguishable from AB (100% score) while the other element B also displays a significant learning effect (50% score). Therefore, the network cannot solve the task of positive patterning. On the other hand, it is more competent with negative patterning. The two individual elements of A and B do get learned and the compound AB is affected but we can still distinguish between the two cases. The same behaviour can be observed for the task of biconditional discrimination where learning the reinforced stimuli again results in the non-reinforced stimuli exhibiting a score as well but the two classes remain separate. Finally, no blocking effect is observed, i.e. element B of the compound AB does not exhibit reduced scores due to prior reinforcement of A.

4.2.2 Robot experiment 1: elemental learning

For the first set of robotic experiments, the robot was tested in the elemental learning paradigm. The two stimuli (A+ B-) were set as completely distinct, i.e. there was no overlap of their input patterns in the first layer of the PNs. Fig. 4.12 - 4.15 show how the response of the neural network evolves as the robot encounters again and again stimulus A, followed by punishment. The plots show the activity of eight neurons. Activity in neuron 1 signifies the presence of the CS (stimulus A) and neuron 5 corresponds to the neutral stimulus B. The CS also activates neurons 2, 3 and 4 and stimulus B neurons 6, 7 and 8 as well but their activity is similar to that of neurons 1 and 5 respectively and they are not included in the plots. The next five neurons (71 to 75) are the ENs which modify their response as a result of a training trial, as described in section 4.1.4. The CR neuron (81) is the last neuron whose activity we have chosen to record. Firing of this neuron "predicts" the US and makes the robot initiate an escape motor response.

Fig. 4.12 shows the response of the network when the robot encounters the CS for the first time, fig. 4.13 is from the second encounter, after some more trials the network responds as in fig. 4.14 and finally, the last trial results in the response of fig. 4.15. As was expected, the initial "naive" response of the ENs gradually changes and becomes weaker (fig. 4.13) until, at some point (fig. 4.14), the CR neuron starts firing, although some of the ENs still generate a few spikes. For the last trial (fig. 4.15), the US is absent (light source turned off) but the

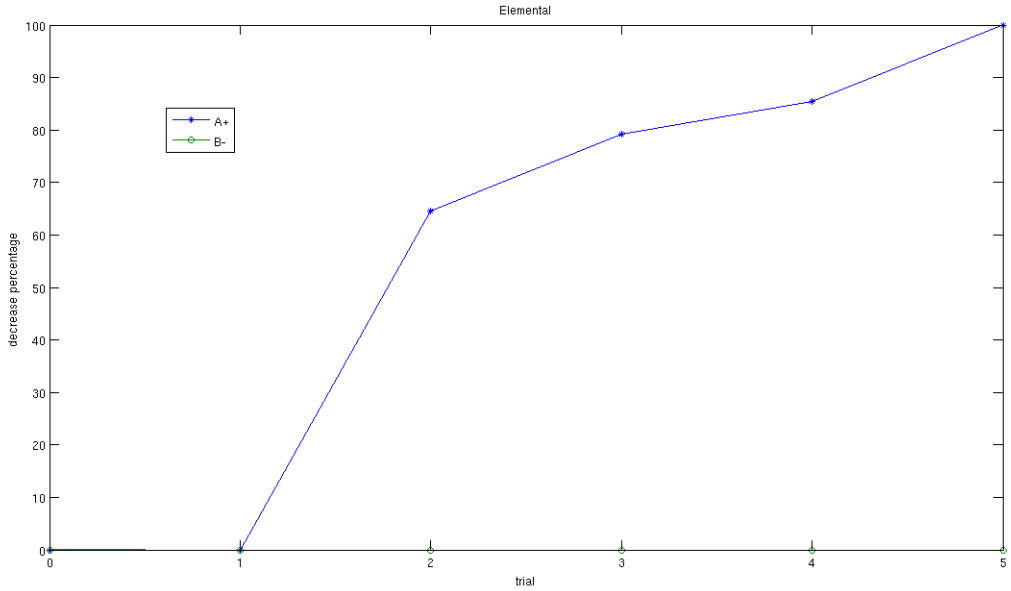


Figure 4.6: Performance of the robot brain (measured as decrease percentage of neural activity, see text) in an elemental learning task for a number of consecutive trials. A+ (blue line) corresponds to the reinforced stimulus, B- (green) to the non-reinforced.

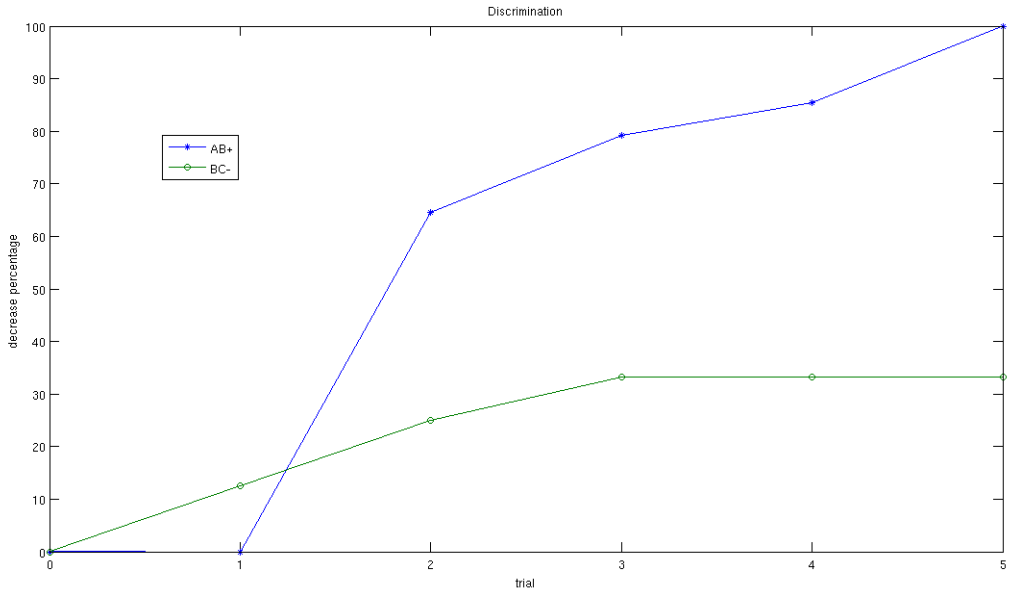


Figure 4.7: Performance of the robot brain in a discrimination learning task for a number of consecutive trials. AB+ (blue line) corresponds to the reinforced stimulus, BC- (green) to the non-reinforced.

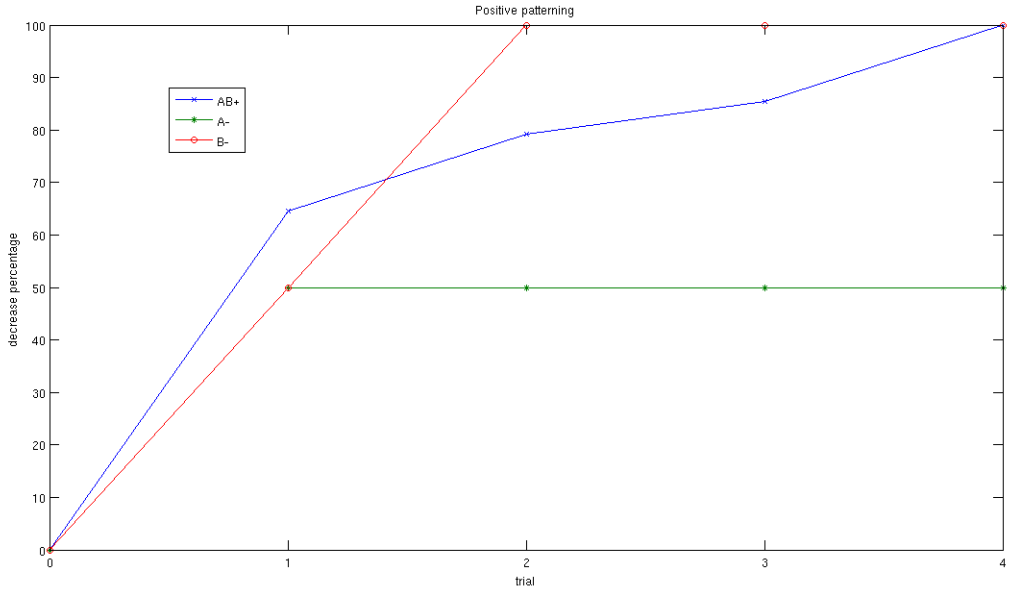


Figure 4.8: Performance of the robot brain in a positive patterning learning task for a number of consecutive trials. AB+ (blue line) corresponds to the compound reinforced stimulus, A- (green) and B- (red) to the separate, non-reinforced stimuli.

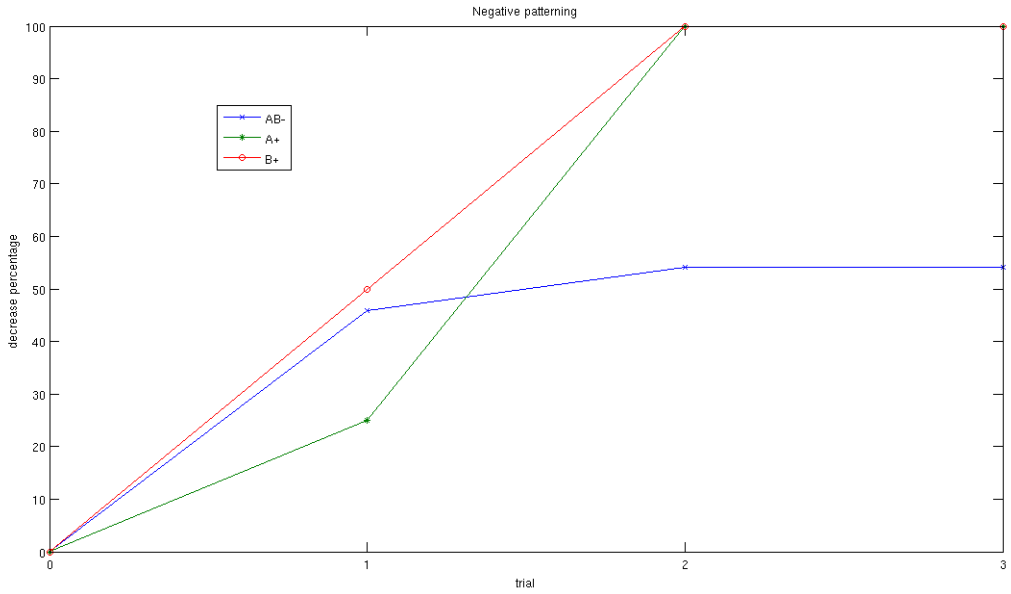


Figure 4.9: Performance of the robot brain in a negative patterning learning task for a number of consecutive trials. AB+ (blue line) corresponds to the compound non-reinforced stimulus, A- (green) and B- (red) to the separate, reinforced stimuli.

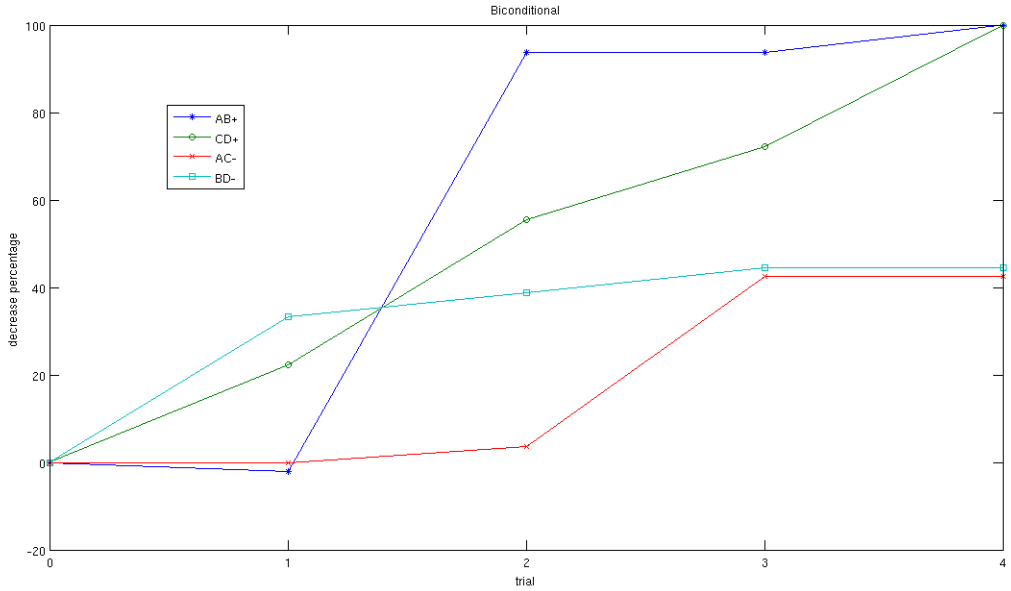


Figure 4.10: Performance of the robot brain in a biconditional discrimination learning task for a number of consecutive trials. AB+ (blue line) and CD+ (green) correspond to the reinforced stimuli, AC- (red) and BD- (light blue) to the non-reinforced.

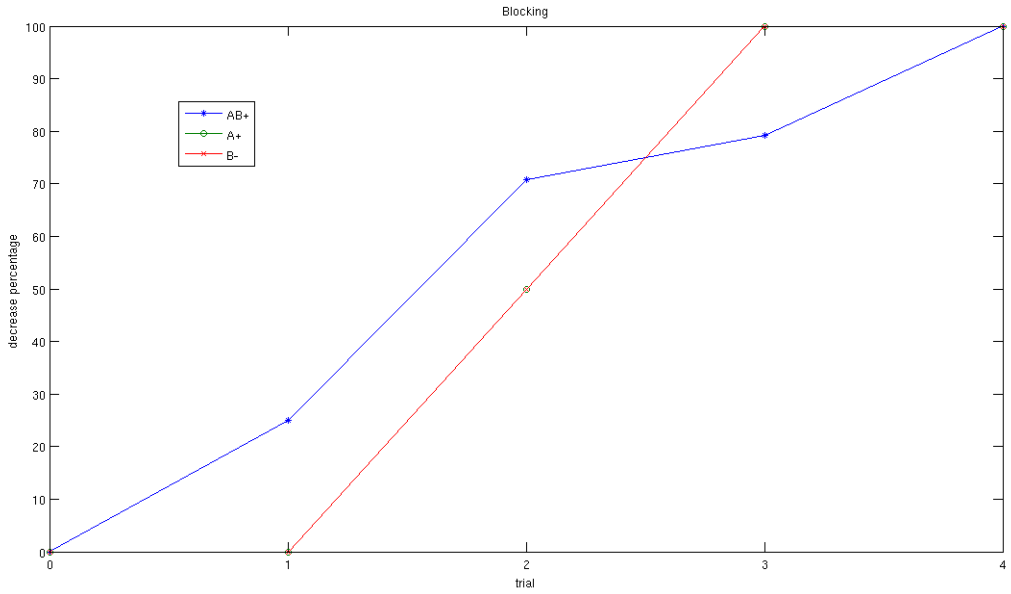


Figure 4.11: Performance of the robot brain in a blocking task for a number of consecutive trials. AB+ (blue line) and A+ (green) correspond to the reinforced stimuli, B- (red) to the non-reinforced.

robot has learned that stimulus A should be avoided, as indicated by the CR neuron. On the other hand, the training procedure has no effect on the response of the network to the neutral stimulus (fig. 4.16 - 4.17). The response of the ENs during the last encounter with stimulus B (fig. 4.17) remains the same as their naive response (fig. 4.16).

4.2.3 Robot experiment 2: discrimination learning

We can make the task more difficult by using stimuli which are more similar to each other. Fig. 4.18 - 4.21 show the behaviour of the network when the two input patterns overlap by 50%. The response of the network to the reinforced stimulus AB before (first trial) and after (last trial) training is depicted in figures 4.18 and 4.19 respectively. Similarly, for the neutral stimulus BC, we can see the effect of learning in figures 4.20 and 4.21. The similarity of the two stimuli does not result in any confusion and robust learning is again observed. There is only a slight interference as far as neuron 75 is concerned (training makes it non-responsive to BC as well) but it does not affect the activity of the CR neuron.

The robot retains the ability to make the correct associations and discriminate between the two stimuli even if we increase the overlap to 75% (fig. 4.22 - 4.25). In fact, in this case the response of the network to BC remains almost unchanged (compare figures 4.24 and 4.25). Although overlapping input patterns necessarily result in overlapping activity patterns in the KCs layer (to a lesser extent than in the PNs), the lateral inhibition among the ENs may act as an additional differentiation mechanism which has the effect of producing quite distinct winning patterns in the ENs layer. Of course, there is also the statistical possibility that sometimes this might work the other way around. Input patterns that are more dissimilar may end up having "converging" ENs responses with increased chances for interference, but this is more like an "artifact" and not a consistent behaviour of the network.

4.2.4 Robot experiment 3: positive patterning

Next, the robot was tested in a positive patterning task. Since this task requires three different odours and our setup has only two "tubes", we first used just one of the tubes in order to train the robot with the reinforced stimulus AB. Subsequently, two different tests were run in which one of the stimulus was always AB and the other was A in the first test and B in the second. As expected, the robot was not able to solve the task. It did learn to avoid AB but it also learned to avoid B as well. Interestingly, stimulus A remained neutral.

This is something that we could have expected too, by looking at fig. 4.26 - 4.29. Stimulus A is not as much affected by the learning procedure as B. Fig. 4.26 - 4.29 show how the experiments affect the responses of the robot brain. Stimulus A remains neutral only because there still is one single spike from the ENs which inhibits the CR neuron and prevents it from

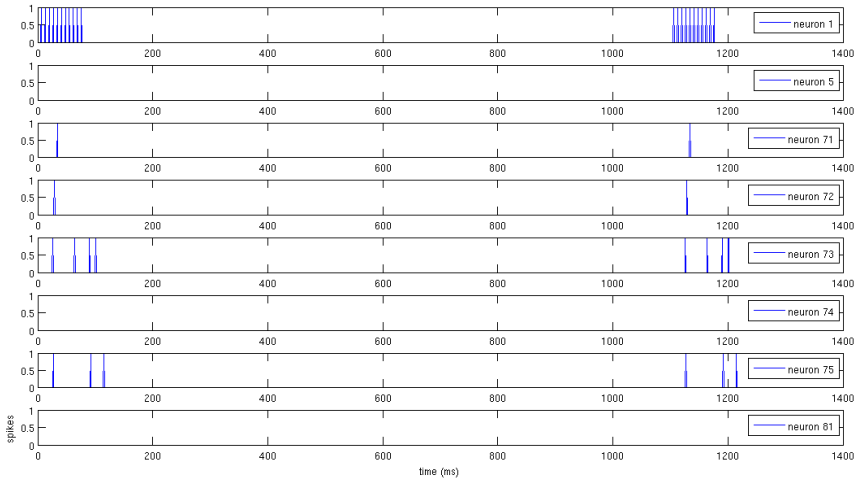


Figure 4.12: Response of the robot brain in an elemental learning task (A+ B-) when encountering the CS (stimulus A) for the first time. Data recorded as spikes during a robot experiment. Neurons 1 and 5 are PNs, corresponding to stimulus A and B respectively, 71-75 are KCs and 81 is the CR neuron.

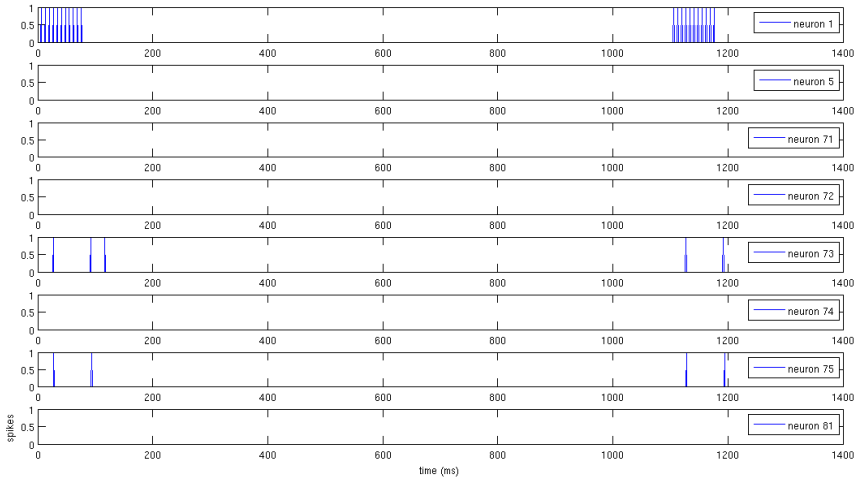


Figure 4.13: Response of the robot brain in an elemental learning task (A+ B-) when encountering the CS (stimulus A) for the second time. Data recorded as spikes during a robot experiment. Neurons 1 and 5 are PNs, corresponding to stimulus A and B respectively, 71-75 are KCs and 81 is the CR neuron.

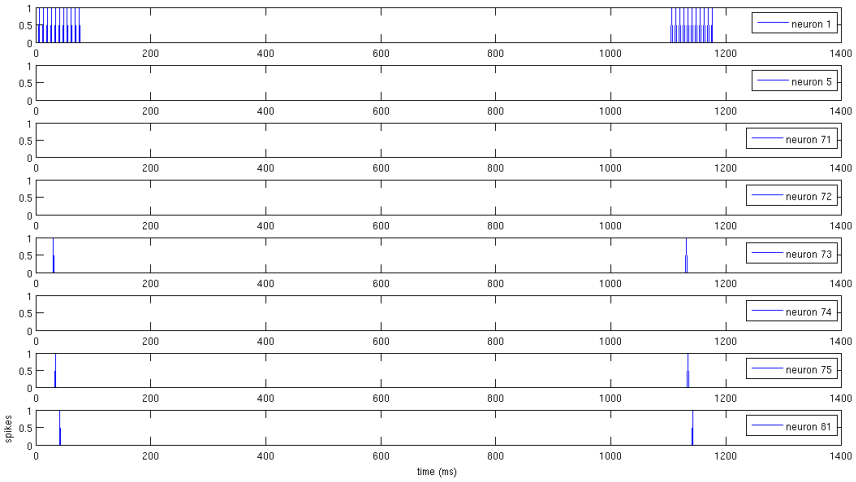


Figure 4.14: Response of the robot brain in an elemental learning task (A+ B-) after several encounters with the CS (stimulus A). Data recorded as spikes during a robot experiment. Neurons 1 and 5 are PNs, corresponding to stimulus A and B respectively, 71-75 are KCs and 81 is the CR neuron.

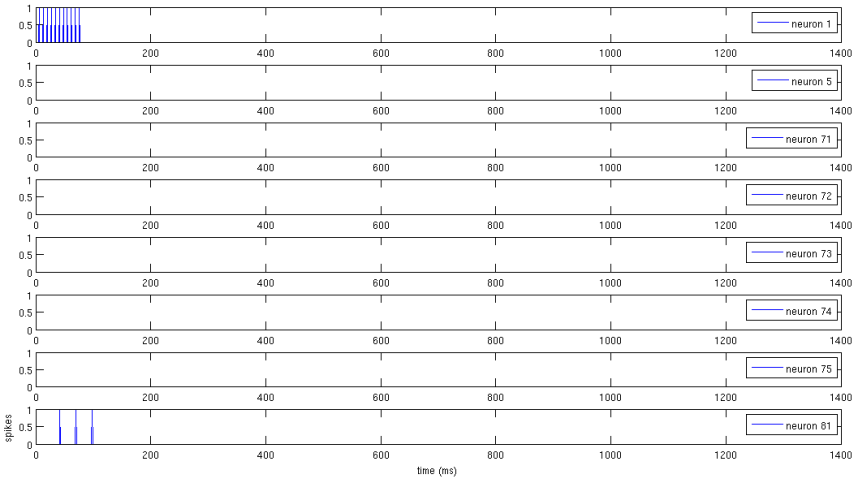


Figure 4.15: Response of the robot brain in an elemental learning task (A+ B-) during the last encounter with the CS (stimulus A). Data recorded as spikes during a robot experiment. Neurons 1 and 5 are PNs, corresponding to stimulus A and B respectively, 71-75 are KCs and 81 is the CR neuron.

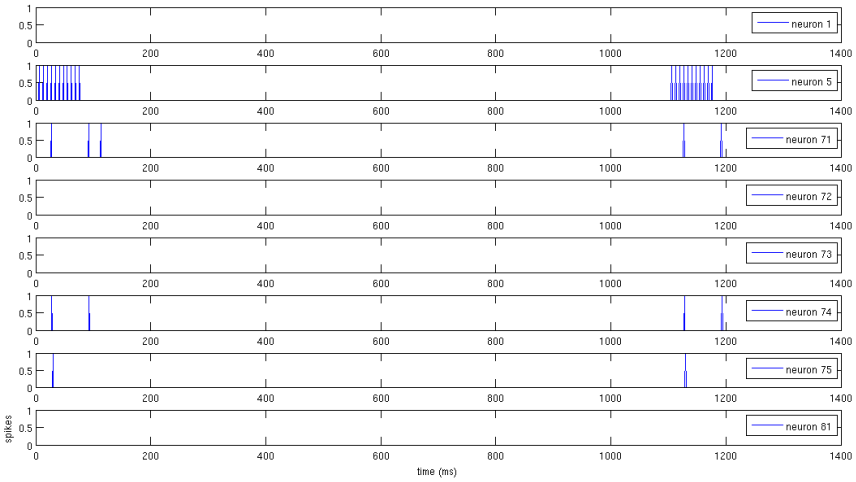


Figure 4.16: Response of the robot brain in an elemental learning task (A+ B-) when encountering the non-reinforced stimulus B for the first time. Data recorded as spikes during a robot experiment. Neurons 1 and 5 are PNs, corresponding to stimulus A and B respectively, 71-75 are KCs and 81 is the CR neuron.

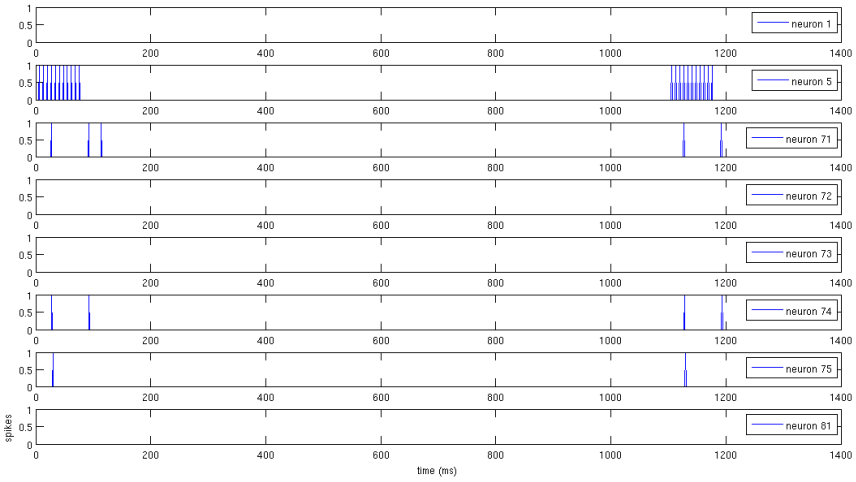


Figure 4.17: Response of the robot brain in an elemental learning task (A+ B-) during the last encounter with the non-reinforced stimulus B. Data recorded as spikes during a robot experiment. Neurons 1 and 5 are PNs, corresponding to stimulus A and B respectively, 71-75 are KCs and 81 is the CR neuron.

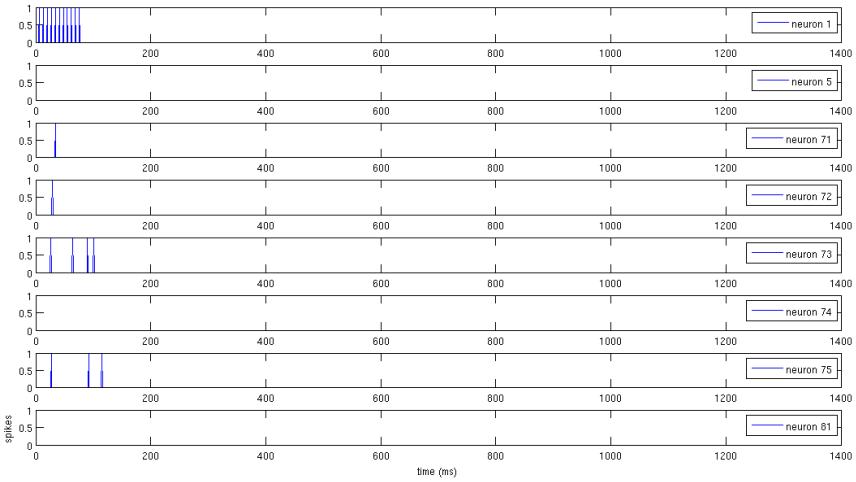


Figure 4.18: Response of the robot brain to AB before training in a discrimination learning task (AB+ BC-, 50% overlap). Data recorded as spikes during a robot experiment. Neurons 1 and 5 are PNs, corresponding to stimulus AB and BC respectively, 71-75 are KCs and 81 is the CR neuron.

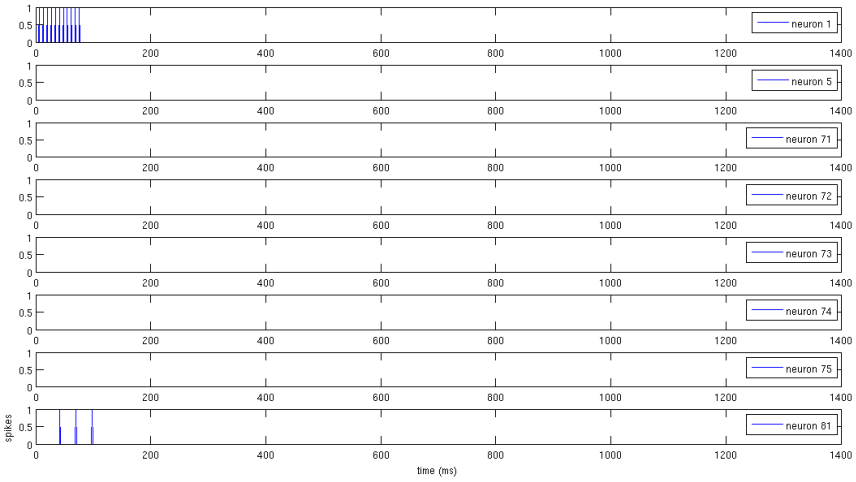


Figure 4.19: Response of the robot brain to AB after training in a discrimination learning task (AB+ BC-, 50% overlap). Data recorded as spikes during a robot experiment. Neurons 1 and 5 are PNs, corresponding to stimulus AB and BC respectively, 71-75 are KCs and 81 is the CR neuron.

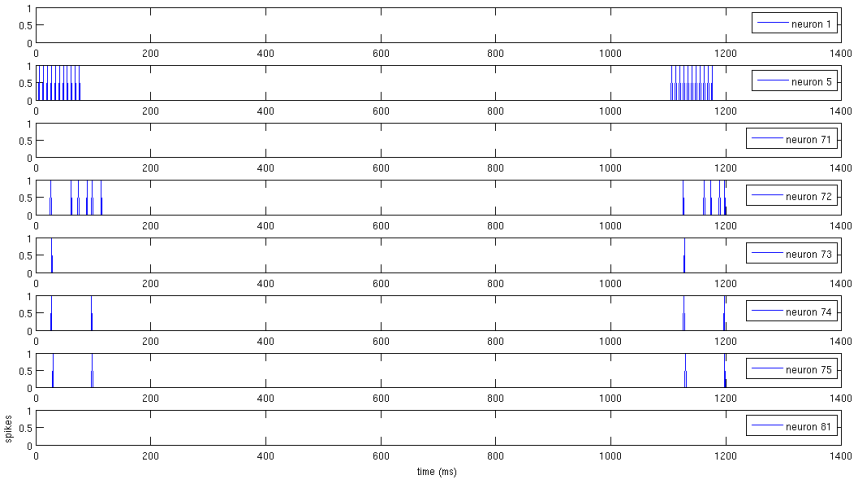


Figure 4.20: Response of the robot brain to BC before training in a discrimination learning task (AB+ BC-, 50% overlap). Data recorded as spikes during a robot experiment. Neurons 1 and 5 are PNs, corresponding to stimulus AB and BC respectively, 71-75 are KCs and 81 is the CR neuron.

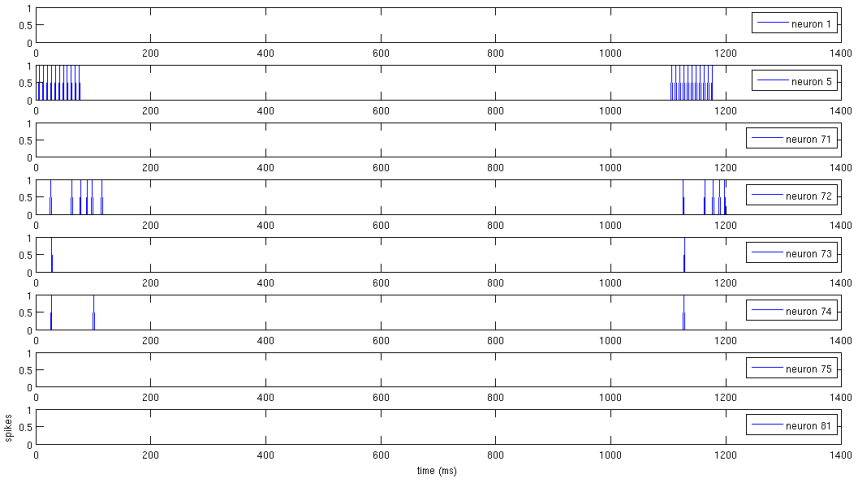


Figure 4.21: Response of the robot brain to BC after training in a discrimination learning task (AB+ BC-, 50% overlap). Data recorded as spikes during a robot experiment. Neurons 1 and 5 are PNs, corresponding to stimulus AB and BC respectively, 71-75 are KCs and 81 is the CR neuron.

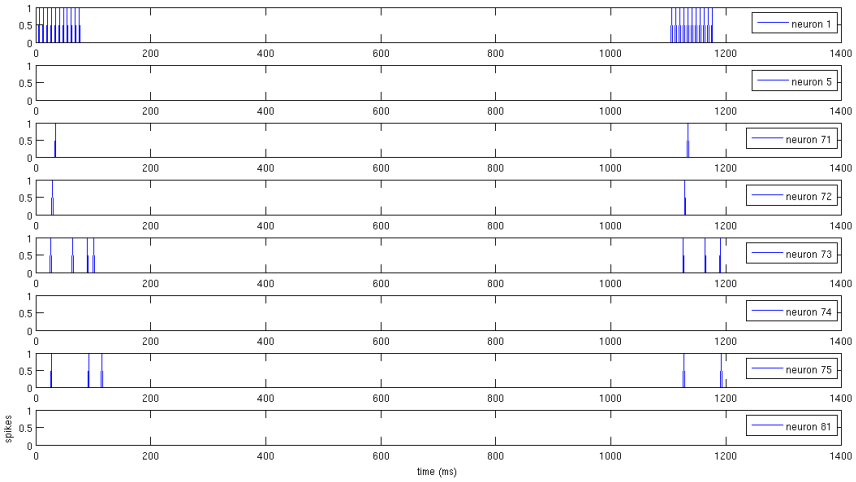


Figure 4.22: Response of the robot brain to AB before training in a discrimination learning task (AB+ BC-, 75% overlap). Data recorded as spikes during a robot experiment. Neurons 1 and 5 are PNs, corresponding to stimulus AB and BC respectively, 71-75 are KCs and 81 is the CR neuron.

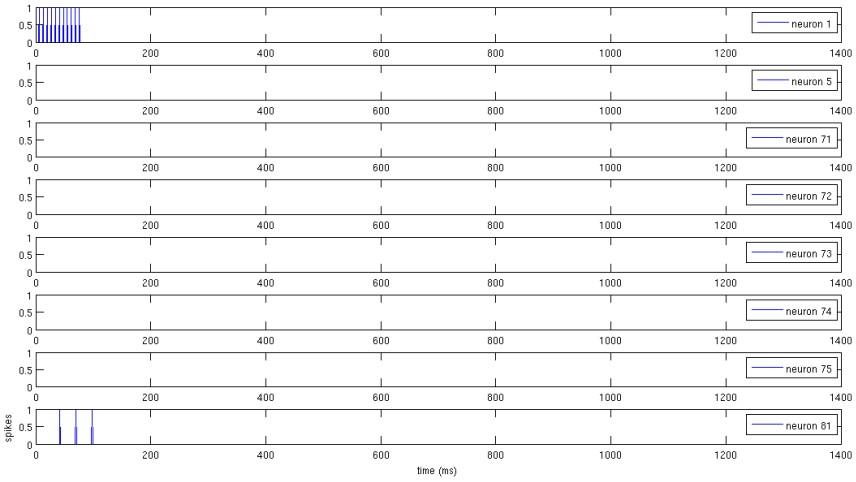


Figure 4.23: Response of the robot brain to AB after training in a discrimination learning task (AB+ BC-, 75% overlap). Data recorded as spikes during a robot experiment. Neurons 1 and 5 are PNs, corresponding to stimulus AB and BC respectively, 71-75 are KCs and 81 is the CR neuron.

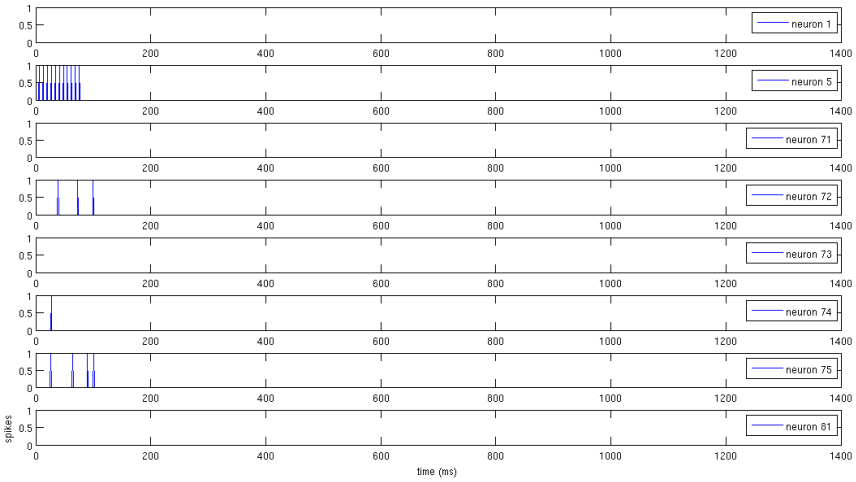


Figure 4.24: Response of the robot brain to BC before training in a discrimination learning task (AB+ BC-, 75% overlap). Data recorded as spikes during a robot experiment. Neurons 1 and 5 are PNs, corresponding to stimulus AB and BC respectively, 71-75 are KCs and 81 is the CR neuron.

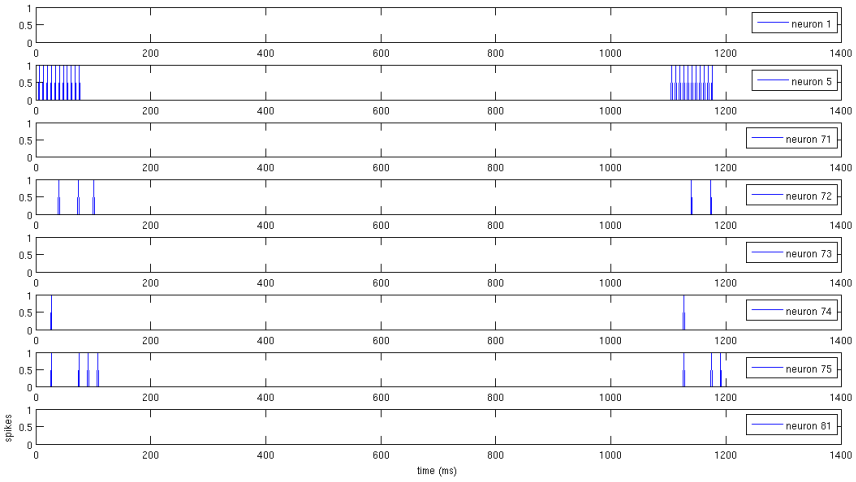


Figure 4.25: Response of the robot brain to BC after training in a discrimination learning task (AB+ BC-, 75% overlap). Data recorded as spikes during a robot experiment. Neurons 1 and 5 are PNs, corresponding to stimulus AB and BC respectively, 71-75 are KCs and 81 is the CR neuron.

generating a sustained spiking activity (fig. 4.28) [4]. Obviously, stimulus A is on the verge of becoming aversive.

4.2.5 Robot experiment 4: negative patterning

For the task of negative patterning, the robot was first trained separately with odour A and odour B and then tested with the choices AB vs A and AB vs B. Fig. 4.30 4.34 show the results. Again, the robot's behaviour is in accordance with the results from the simulation (fig. 4.9). We can see that the robot is much more efficient at solving negative than positive patterning. The two classes of stimuli (reinforced A, B and non-reinforced AB) are separated quite distinctly. If we compare the response of the network to AB after it has been trained with A and B (fig. 4.34) with the "naive" response to AB (fig. 4.26), it is clear that it is only slightly affected by the learning process.

4.2.6 Robot experiment 5: CS without a "refractory" period

For the last experiment, we decided to modify the way the CS is fed to the neural network. In order to achieve a "correct" timing for the neuromodulation mechanism to work, whenever the robot encounters the CS, the input pattern corresponding to the CS is presented to the network only for a limited amount of time, followed by a "refractory" period. During this period, the CS is not fed to the network, although the robot might still sense it. As explained in section 4.1.5, the presentation of the US follows a similar logic. By limiting the time windows during which the CS or the US are effective (from the perspective of the neural network), we can treat them as "momentary" phenomena, similar to the way most, purely software, studies treat them.

Cancelling the refractory period of the CS brings us closer to a more realistic environment. After all, the odours in a T-maze are not presented to the flies only for a brief moment. On the contrary, the odours are always present while the electric shocks are delivered in regular time intervals. Fig. 4.35 shows the response of the network to the presentation of the CS for the first learning trial. It is obvious that the robot learns to associate the CS with the US after just one trial. In fact, it was observed that during this first trial, the robot gets punished not only by the US but subsequently by the reinforced CS. Section 4.3.3 discusses in more detail the possible interpretations of this behaviour.

4.3 Discussion

An objection which could be raised against the whole undertaking of this chapter is that it does not exactly come up to our initial expectations. What more have we learned out of this biorobotics approach? Was it really worth the effort of actually building a physical, behaving

[4] An escape response is initiated only if the spiking frequency is higher than 2 spikes/100 ms

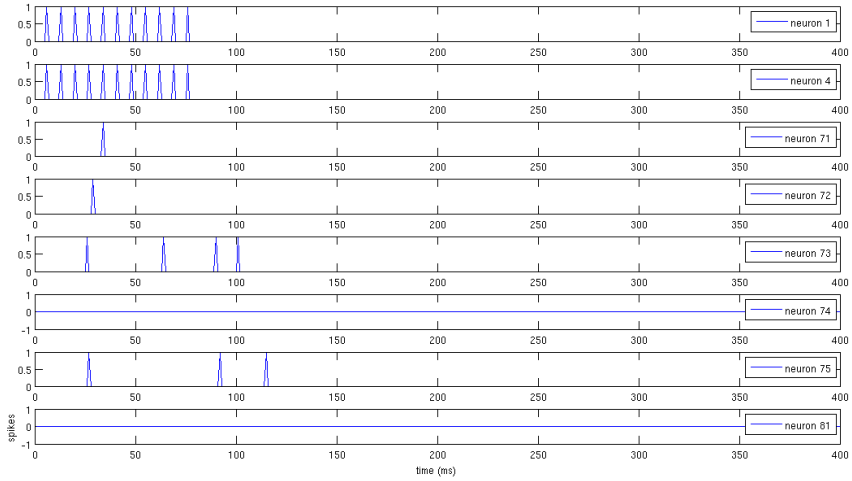


Figure 4.26: Response of the robot brain to AB before training in a a positive patterning task (AB+ A- B-). Data recorded as spikes during a robot experiment. Neurons 1 and 5 are PNs, corresponding to stimulus A and B respectively, 71-75 are KCs and 81 is the CR neuron.

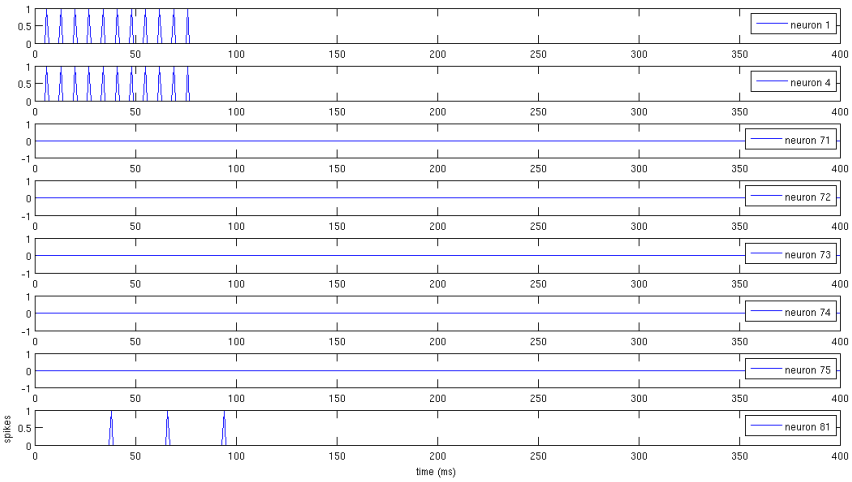


Figure 4.27: Response of the robot brain to AB after training in a a positive patterning task (AB+ A- B-). Data recorded as spikes during a robot experiment. Neurons 1 and 5 are PNs, corresponding to stimulus A and B respectively, 71-75 are KCs and 81 is the CR neuron.

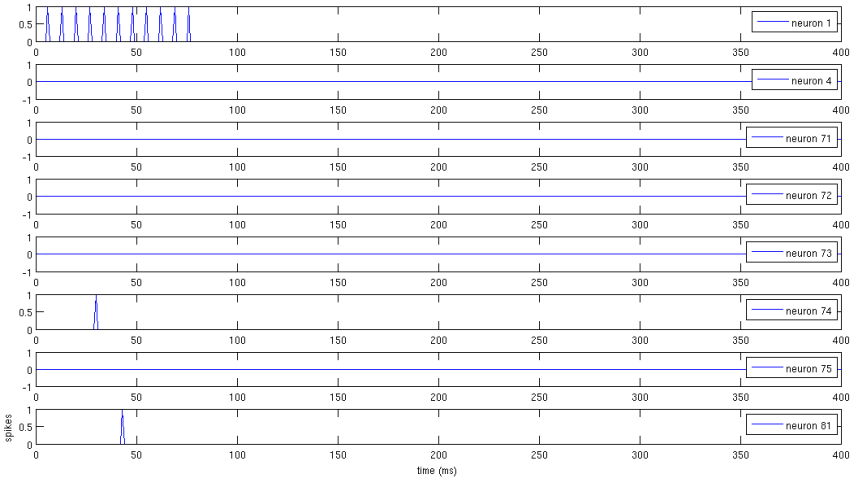


Figure 4.28: Response of the robot brain to A during testing in a a positive patterning task (AB+ A- B-). Data recorded as spikes during a robot experiment. Neurons 1 and 5 are PNs, corresponding to stimulus A and B respectively, 71-75 are KCs and 81 is the CR neuron.

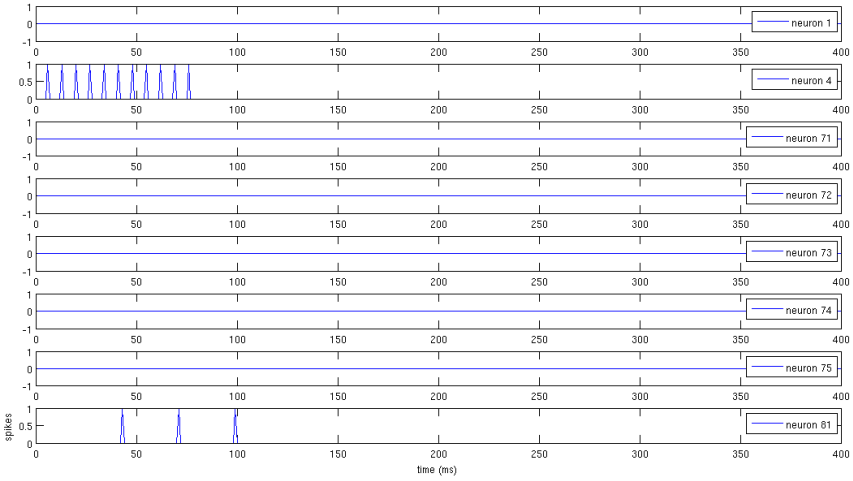


Figure 4.29: Response of the robot brain to B during testing in a a positive patterning task (AB+ A- B-). Data recorded as spikes during a robot experiment. Neurons 1 and 5 are PNs, corresponding to stimulus A and B respectively, 71-75 are KCs and 81 is the CR neuron.

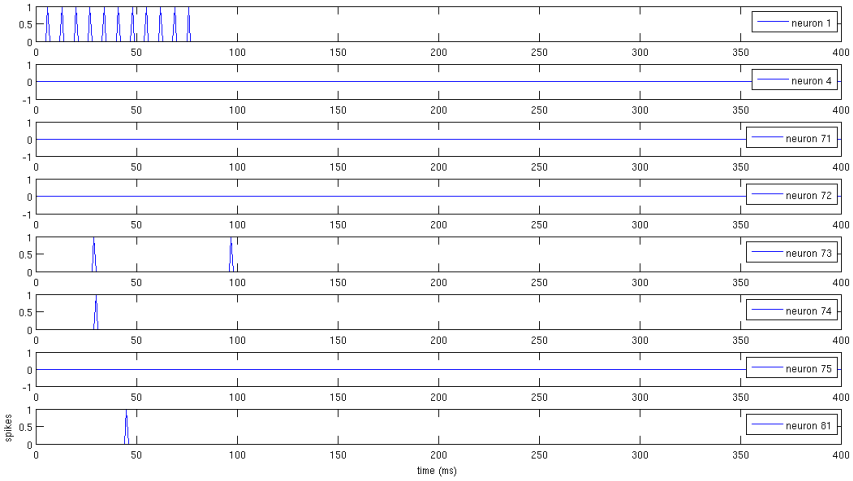


Figure 4.30: Response of the robot brain to A before training in a a negative patterning task (A+ B+ AB-). Data recorded as spikes during a robot experiment. Neurons 1 and 5 are PNs, corresponding to stimulus A and B respectively, 71-75 are KCs and 81 is the CR neuron.

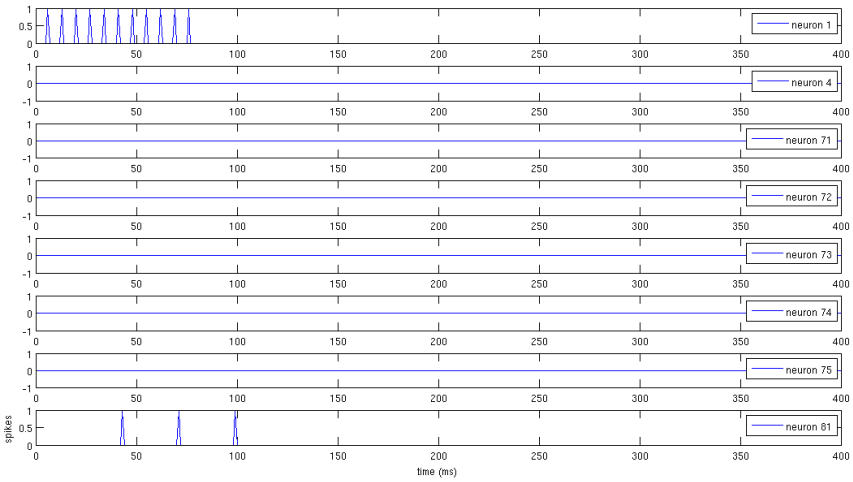


Figure 4.31: Response of the robot brain to A after training in a a negative patterning task (A+ B+ AB-). Data recorded as spikes during a robot experiment. Neurons 1 and 5 are PNs, corresponding to stimulus A and B respectively, 71-75 are KCs and 81 is the CR neuron.

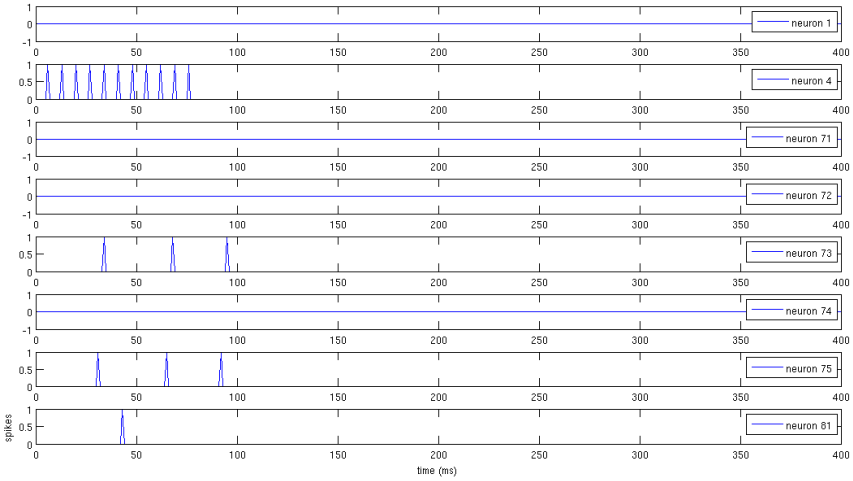


Figure 4.32: Response of the robot brain to B before training in a a negative patterning task (A+ B+ AB-). Data recorded as spikes during a robot experiment. Neurons 1 and 5 are PNs, corresponding to stimulus A and B respectively, 71-75 are KCs and 81 is the CR neuron.

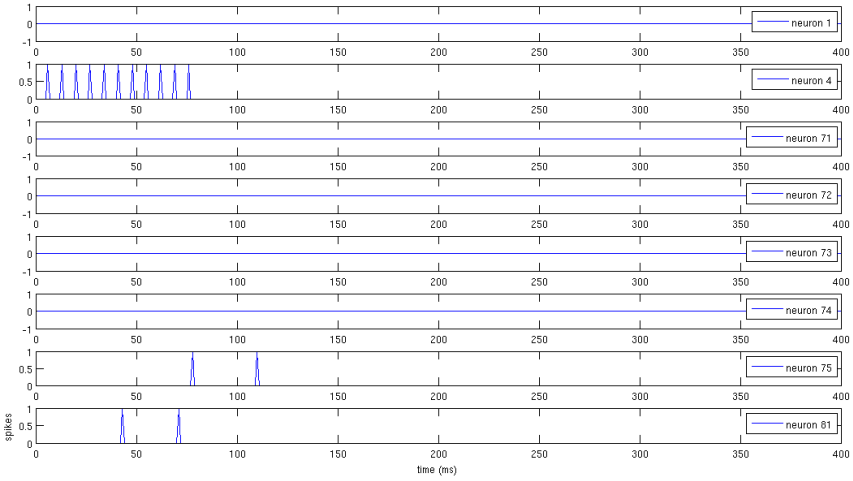


Figure 4.33: Response of the robot brain to B after training in a a negative patterning task (A+ B+ AB-). Data recorded as spikes during a robot experiment. Neurons 1 and 5 are PNs, corresponding to stimulus A and B respectively, 71-75 are KCs and 81 is the CR neuron.

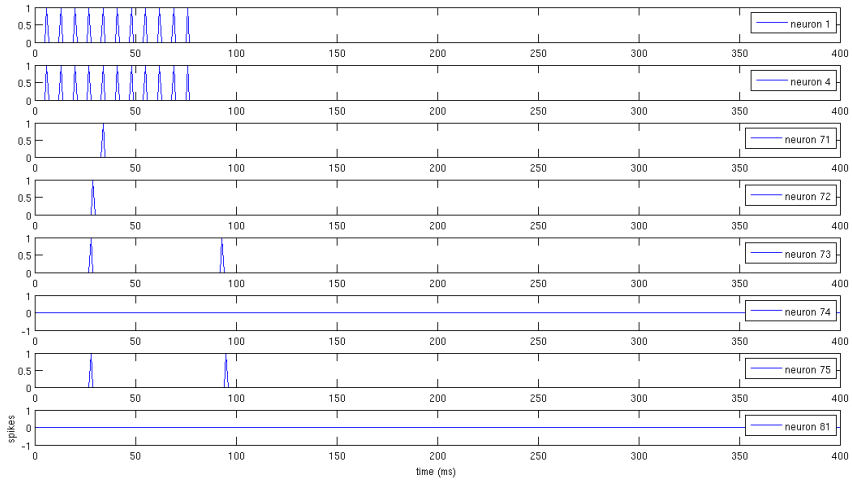


Figure 4.34: Response of the robot brain to AB during testing in a a negative patterning task (A+ B+ AB-). Data recorded as spikes during a robot experiment. Neurons 1 and 5 are PNs, corresponding to stimulus A and B respectively, 71-75 are KCs and 81 is the CR neuron.

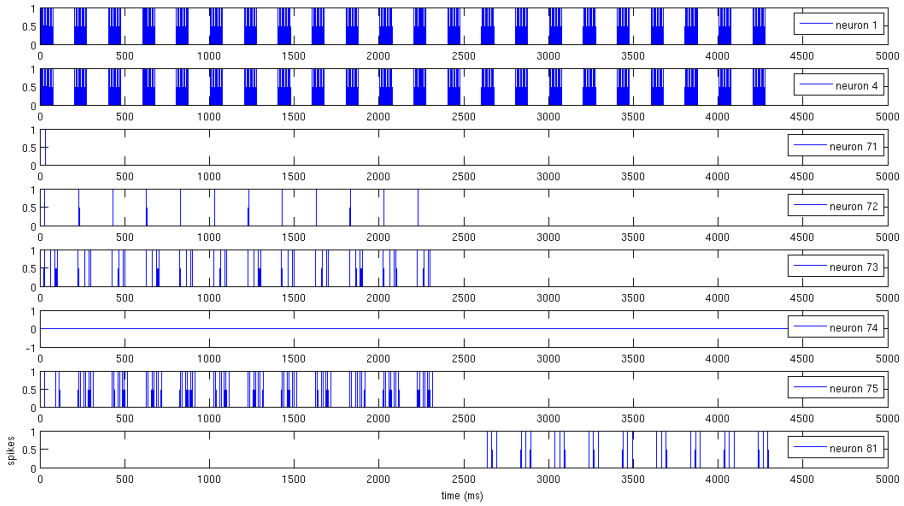


Figure 4.35: Behaviour of the robot brain when a CS is presented to it without a refractory period

agent? After all, the simulations of the network on the FPGA had already shown us its limits and capabilities while using it on a robotic platform now seems as a nice way to torture ourselves. Although it might have some interesting applications on the implementation of more intelligent robotic guidance systems, it seems doubtful that it can offer any insights to basic science. This objection cannot be discarded as simply wrong or irrelevant. Indeed, the process of building the biorobotic platform is more like an attempt towards a straightforward implementation of what was already known than an attempt to answer questions by watching the behaviour of the neural network. Our robot and its brain carry the burden of too much previous knowledge.

On the other hand though, the perception of the implementation process as being simple and straightforward is not very accurate. We have chosen to present our work in the traditional manner where one provides simple, step-by-step instructions, so that others may be able to repeat it, without mentioning any obstacles or difficulties that appeared. Although this might be necessary in order not to obfuscate the whole process with too many details and confuse the reader, it should be noted that this is exactly where one of the strengths of the biorobotics approach lies, namely that all these various obstacles force the experimenter be fully aware of what assumptions have been made. Therefore, there is room for theoretical discussion, maybe not based directly on the experimental results but on the process of actually setting up the arena and the robotic system. This is where the concept of negative or liminal scientific knowledge might apply (Cetina, 1999).

4.3.1 Some remarks with regard to the limitations of the robotic platform

Considering the technical difficulties of building the platform, it should not come as a surprise if we say that a straightforward implementation was intended, to some extent. VHDL is surely a seductive language which gives the impression that designing a digital circuit is just a matter of writing some hundreds of source code lines. This is not an inaccurate impression, as long as we restrain ourselves to relatively simple designs which the software tools can easily synthesize, map and route. When we move to more complex designs, it becomes clear that some of the programming conveniences which VHDL offers may very well turn into traps, when used recklessly (and the temptation for convenient solutions is of course quite strong for inexperienced designers) and that designing in a bottom-up manner proves much less error-prone in the long-term (where by “bottom-up” design it is meant that the designer thinks in hardware terms and then tries to implement the hardware architecture with the help of VHDL and not the other way around).

Moreover, integrating the design within an embedded system and then connecting the whole system to the robot is not a trivial task. Even some insignificant details (like the inability of the FPGA to implement a serial port controller with a hardware buffer longer than

16 bytes while the robot may sometimes respond with messages greater than 16 bytes in size) can offer plenty of hours and days of debugging "fun", as every roboticist knows. For these reasons, it was imperative that we first have a working system before moving to something more complicated. The robotic platform presented in this chapter should thus be considered as a proof of concept, a preliminary but working prototype which can later be amended in order to run more meaningful (from a neurobiological point of view) experiments, suitable for generating a more positive kind of knowledge.

Being a prototype, the platform has certain limitations which restricted the range of experiments that we could run. For example, the 16-bit addressing mode of the simulator prevented us from simulating networks with more than some hundreds of neurons. Even more important was the fact that we had to use the less powerful FPGA board for our experiments with the robot, as a result of the need for two serial ports. An even smaller network was thus implemented for controlling the robot. Moreover, once synthesized, the network could not be modified, neither with respect to its parameters nor to its architecture. As a consequence, we had very limited flexibility as far as the range of "brains" which we could study is concerned. Finally, the use of the IR and ambient light sensors provides us with a very poor (in terms of available stimuli) environment in which the robot could be tested. Section 5.1 in the next chapter discusses how these limitations could be overcome.

4.3.2 Comparing the robot brain to the insect brain

Leaving aside for the moment the robot itself, it is interesting to take a look at the properties and capabilities of its "brain" (section 4.1.4). The results from the simulation and the experiments indicate that the theory of associative learning which is more compatible with them is the theory of configural cues (section 2.1). There is a significant difference though. The theories of associative learning which are based on the Rescorla-Wagner rule predict that learning may occur in every presentation of a stimulus, even if no US is present. Usually, these US-less trials result in a decrease of the associative strength of the stimulus.

Contrary to this, our neural network cannot modify any synaptic weights without a US. It functions more like a straightforward separator. A compound AB has a separate representation of its own among the KCs (this is the main function of the MB) and this is the reason why the network can solve the negative pattering task. The KCs which represent the individual elements A and B "lose" their synaptic connections to the ENs but the extra KCs which correspond exclusively to the compound AB remain intact. On the other hand, the positive patterning task is much harder to solve because reinforcing the compound AB results in a widespread synaptic loss that affects both the compound and the elements.

If we make the winning ENs pattern for every stimulus very distinct from that of every other (so that the pattern generated by AB would be much more different than those by A or

B), then positive patterning would become much easier. We could probably achieve this with certain modifications, like increasing the network size, introducing lateral inhibition among the KCs or adding a gain control mechanism in the first layer of the PNs. However, this would be equivalent to implementing the extreme configural cue theory according to which each stimulus has an independent representation (AB is completely unrelated to A or B).

The issue becomes even more complicated and intriguing when we compare the performance of our tiny brain with the learning scores of insects in actual behavioural experiments. A recent comprehensive study of the flies' learning abilities (Young et al., 2010) has shown that their performance in non-elemental learning problems (negative patterning, biconditional discrimination) is very poor. Does this mean that a tiny brain with 81 neurons is more powerful than the fly brain with its 2500 neurons only in the MB? It could be argued that our model has included too many assumptions which tune it to this specific problem whereas the real fly brain has a substantially different architecture, connectivity and plasticity mechanisms. It is also plausible that the MB serves a different function than what has been accepted until now. However, an older study (Deisig et al., 2001) found that bees are capable of solving both negative and positive patterning tasks. Of course, it might simply be a matter of size, since the bee brain is significantly larger than the fly brain, but our data show that brain size is unlikely to be a crucial factor, at least as far as these relatively simple tasks are concerned.

Another possible reason for this discrepancy might have to do with the experimental setup and procedure. Flies were trained in the T-maze and the US consisted of electric shocks (aversive conditioning) whereas appetitive conditioning of the proboscis extension reflex was used for the bees. It might be the case that aversive conditioning (especially shocking) induces some kind of learned helplessness which renders flies incapable of solving any complex tasks while appetitive conditioning "motivates" them to learn. In fact, there are experimental data which support the idea that learned behaviour depends on outcome expectations (Gerber and Hendel, 2006).

Of course, this discussion naturally leads us to notions such as "attention" and "motivation". While conceptually relevant and meaningful, these are usually employed within frameworks which remain fundamentally cognitivist. They are mostly seen as external parameters, functioning as gates for the input stimuli, but the basic machine metaphor for the brain remains unchallenged. At the same time, they give off the scent of a homunculus argument. How is attention controlled and modified? Do we need another, higher attention for this lower attention? Maybe something even more radical is required, like "intentionality" and "embodiment" (Freeman, 2007). Maybe we need to reconsider some older and forgotten concepts, such as the concept of "intention" (Nunez and Freeman, 1999). In any case, it seems that we need to incorporate top-down processing mechanisms in our models in order to account for the results we get from behavioural experiments.

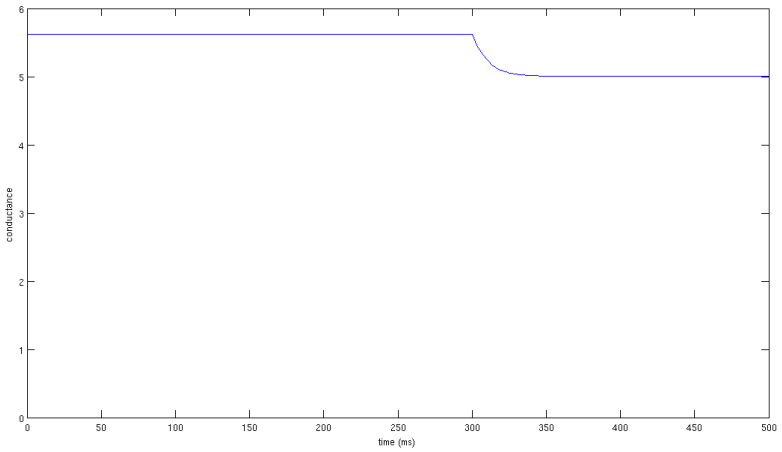
4.3.3 The role of STDP

The learning rule that we used is a modified form of STDP. Essentially, it is just like STDP but the difference is that the synaptic modifications get registered only if a neuromodulator release follows after some seconds. The choice to use this modified form of STDP was made because the simple STDP rule was considered unsuitable. During the initial stages of building the robotic platform, some experiments were run with a robot brain whose learning mechanism was the simple STDP rule. The presynaptic neurons were activated by one stimulus A and the postsynaptic by another stimulus B which was expected to be associated with A. However, in order to make STDP functional during robot operation, we had to slow down the neural network so that seconds of robot behaviour corresponded to milliseconds of brain activity. It has been argued that the causality exhibited by STDP might reflect the causality of external events (Dan and Poo, 2004). Considering the time scales at which STDP works, this seems very doubtful. Additional, possibly quite elaborate, neural structures would be required, with the ability to guide the signals of the stimuli to be associated to the correct place (pre- and post-synaptic neurons) and with the correct timing. For this reason, simple STDP was rejected as a rule that can link learning behaviour to neural plasticity mechanisms.

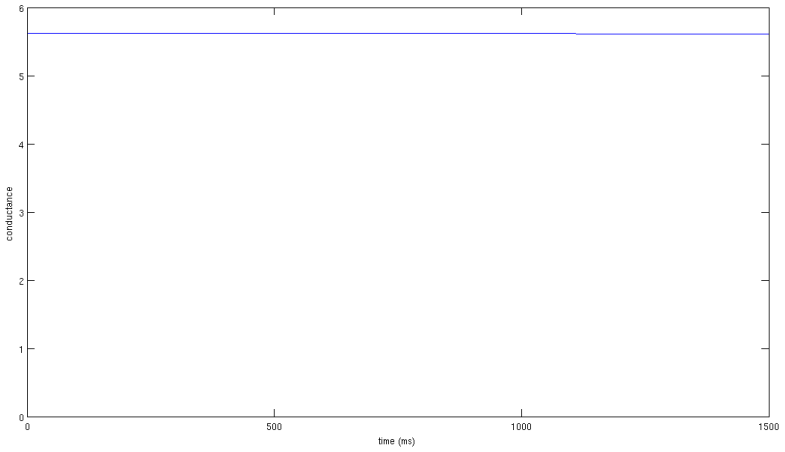
The so-called distal reward problem can be solved by the modified form of STDP (Izhikevich, 2007), as our experiments also show. The effective time range of the learning rule can now span over several seconds but there still remains a problem. The stimuli, both the CS and the US, have to be presented to the network as distinct, almost instantaneous inputs and this is why we have introduced a "refractory" period after stimulus onset in the experiments (section 4.2.6). It is highly unlikely that animals actually perceive the stimuli of their environment in such a way. The purpose of the experiment described in section 4.2.6 was to investigate the behaviour of the robot when the CS is presented to it in a more realistic manner. As we saw, the robot was able to "overlearn" the CS in just a single trial.

At first glance, this might not seem as a very exciting or even interesting finding. The CS is simply closer in time to the US and the learning effect is naturally more pronounced. Fig. 4.36 indicates that the difference between the two cases might have more serious implications. When we tried to modify the STDP parameters so that the learning rate drops to a more normal value, it was found that the A_- parameter (see section 3.1.2) had to be decreased by two orders of magnitude, from the value of 0.2 to 0.002. Fig. 4.36(a) depicts how the synapse from neuron 22 to neuron 73 is weakened when $A_- = 0.002$ and the CS is constantly fed to the network, even at the time of the neuromodulator release. The decrease percentage is about 10%. Using the same value for A_- and presenting the CS instantaneously (only for 100 ms), followed by the US after 1 second, the synaptic modification is negligible (fig. 4.36(b)). Therefore, STDP is unable to function in both modes.

If STDP is so sensitive to timing issues, then its link to behaviour might be less direct than



(a) CS without a refractory period



(b) "Instantaneous" CS

Figure 4.36: Modification of synapse projecting from neuron 22 to neuron 73

previously thought. For example, instead of functioning as a learning mechanism, it might serve an auxiliary role and function as an "attention" or "priming" mechanism. We could imagine a neural pathway starting with a filtering stage (dependent on top-down expectations) which lets the novel stimuli pass through (AL?). Subsequently, these stimuli go through any required processing (MB?) before reaching a "final" stage (ENs), via STDP synapses. In this case, STDP could function as an attention mechanism by increasing the strength of those synapses whose source neurons fire more persistently. The STDP timing windows could also be of significant importance here (fig. 3.1). Besides synaptic conductances, the shape of these windows might be modified as well so that the longest and highest of them correspond to the most salient stimuli. The target neurons of these STDP synapses would then use the time window in order to "recruit" some of the source neurons. A recent study has confirmed that neuromodulators have the ability to influence the STDP time window (Zhang et al., 2009). Such a pathway should therefore be responsible for generating a short-term memory "trace", according to which stimuli are novel and how much attention they "deserve" and it should be mostly active during acquisition. Actual storage of memories might take place in other structures, outside the MBs or even in other lobes of the MBs (see also section 2.3.3). Although the pathway described here is not currently supported by any neurobiological data, it provides a framework for rethinking the role of STDP. Of course, before embarking on a project to reinvent STDP, it would be wise to examine other, less "exotic" solutions to its timing problems, e.g. the existence of additional, compensating structures that bring strong (long-lasting) and weak (brief) stimuli closer cannot be excluded.

4.3.4 Stimulus or "stimulus"?

As a final note, we would like to make some comments on the concept of stimulus. From the discussion about the robot's control algorithm (section 4.1.5), it is evident that the robotic platform has been built with the assumption that a stimulus follows a unidirectional path, from the sensory stage to the neural network and finally to the motor stage with these three processes functioning independently. Although this might be a convenient way to build robots (or at least prototypes), it is not at all obvious that this is also the case for the fly brain. The issue of how to connect the neural network with the sensory input raises a number of questions whose answers are taken for granted in many modelling efforts.

One such crucial question concerns the very concept of the stimulus itself. Just like every concept, it acquires its meaning only in relation to a system of other concepts which together constitute a mental framework, implied in every utterance of the word "stimulus". This framework presupposes a clear-cut distinction between a living organism and its environment and an external world neatly organized in clearly defined stimuli which the organism simply has to internally represent. Learning is thus the internal (re)arrangement of these little, tasty chunks of

external world or, in other words, the establishment of associations among them. This approach might not be problematic when we have to deal with very well controlled environments where we know beforehand all the possible stimuli, such as an arena for robot experiments. It is not clear how it could work in a more realistic environment, full of uncertainties and ambiguities. Moreover, the presence of that sneaky homunculus argument can be felt again. This time, the role of the homunculus has been reserved for the experimenter himself.

We could then attempt to examine the active role that an organism plays in actually shaping its own experiences, even at the perceptual level. Although this is not a new idea (Gibson, 1979), its implications have not been fully explored, possibly due to the fact that the theoretical background for modelling studies is yet not solid enough and that the available tools of neuroscience are not as refined as they should for running the required, delicate experiments. However, it seems to be gaining some additional momentum lately (Engel et al., 2001), (Churchland et al., 1994). It is thus not unreasonable to assume that what an organism "perceives" might depend significantly on what it "remembers". If this is the case, then future models should probably have to explicitly incorporate such top-down processes. Expressing this in more abstract and somewhat philosophical terms, experience can be understood as some kind of dialectical resonation between the environment and the organism which constantly strives to achieve a maximal grip on this environment (Dreyfus, 2002). From this perspective, perception cannot be separated neither from memory nor from motion. This does not imply that all the aspects of an organism's behaviour are identical or that we cannot decompose it in order to study its parts separately. It simply means that they form a continuum, a "seamless fabric" which makes it hard or even impossible to fully understand the parts without the whole.

> *Perception is the process in which certain parts of the environment that are defined by dynamically changing receptors are joined into the structure of the organism-environment system. Perception is a process involving the whole organism-environment system.*
>
> *...*
>
> *Perception is not a linear process proceeding from the stimulus to the percept, but, rather, a circle involving both the sensory and motor organs as well as the events in the environment. A perceptual process does not start with the stimulus, rather the stimulus is an end of this process.* (Jarvilehto, 1999)

Chapter 5

Conclusions and further work

We conclude this thesis with a discussion about the possible future research paths and with a summary of what has been achieved until now.

5.1 Next steps

As discussed in the previous chapter, the robotic system presented in this thesis is just a prototype. Some of its limitations are not that important while some others definitely need to be addressed. Improvements or even radical redesigns are deemed necessary, if we are to run more complex experiments. This section presents the changes that could substantially improve our platform and the experiments that a new setup would allow us to run.

5.1.1 Improving the interface

One of these limitations concerns the programming weaknesses of the MATLAB environment itself. MATLAB was chosen as the platform with which the GUI was built mainly due to its plotting and data processing capabilities. However, from a certain point of view, it resembles FPGAs in that it is quite efficient for rapid prototyping but it does not respond equally well when heavier programming demands are to be met. For example, the lack of any threading capabilities which is an important feature when dealing with robot control, forces the developer to program in an unnatural way (timers were used in order to somehow simulate threads). This situation exacerbates the problems that arise out of the general unwillingness of MATLAB to follow software engineering rules, resulting in systems without any obvious architecture, looking more like a random collection of unrelated pieces of source code. Moreover, MATLAB does not excel as far as its speed performance is concerned, with the newly introduced object-oriented features (useful when trying to impose an architecture) making things even worse. For these reasons, MATLAB is no longer considered appropriate and should be replaced by another environment, such as Python (which is also free and open source, as opposed to MATLAB)

for the interface and communication with the FPGA and some C/C++ libraries for the more computationally intensive simulation engine.

A similar problem was encountered with the software running on the Microblaze processor. This software application is responsible for controlling the robot (sending motor commands and reading the sensors), for sending the recorded data back to the GUI and for updating the state of the neural network, functions which are more easily conceived as parallel processes. Since the application is written as a single, monolithic program, we have again the problem of not being able to assign these functions to different threads. Therefore, the future versions of our system should make use of the *xilkernel* OS, a tiny operating system, developed by Xilinx (or any other operating system, in case another FPGA platform is preferred).

Running the application from within an operating system will not only enable us to use the *Pthreads* standard but will make interfacing with an Ethernet controller easier. In turn, an Ethernet connection will allow for increased data transfer rates (e.g. 100 Mbps) and a closer examination of the neural network's activity during the robot experiments. Another, even more important reason for connecting with the GUI via Ethernet is that we will then be able to implement our system on the XC3SD3400A (or an even more powerful) FPGA.

5.1.2 A more flexible simulator

As shown in fig. 3.13, we have implemented networks of only up to 306 neurons in size with the XC3SD3400A FPGA. It has to be noted that this upper limit was not imposed by the FPGA itself, i.e. by a lack of hardware resources. In fact, we can see that the network with the 306 neurons consumes only 10-12 RAM blocks (fig. 3.13(a)) whereas the FPGA has a total of 126 RAM blocks. It is due to our design that we cannot implement networks with even more neurons (depending, of course, on the synapses to neurons ratio). More specifically, the RAM blocks of the design are organized in words of 16 bits, a width that limits the range of addresses that can be pointed by the synapses' indices (see 3.2.1). Therefore, if we modify the design so that its RAM blocks are organized in 32-bit words (something which has already been done), it is quite plausible that the XC3SD3400A FPGA will be able to accommodate a complete (or almost complete) model of the fly's olfactory pathway (about 2500 KCs, see 2.2).

Besides this slight modification of our digital system, work towards a more fundamental redesign has already begun. There are many reasons for this decision. One of them is the need to have a more complete control over the design by simplifying it (together with the VHDL source code) and getting rid of any redundancies which result in an inefficient utilization of the hardware resources (see also the discussion in 4.3). Each core of the current design is built around an arithmetic unit and a finite state machine which feeds the unit with the correct inputs until the new value of a specific variable has been computed (see section 3.2.1 and fig. 3.4). This solution might work well when only a few variables (and therefore equations) need to

be updated but it becomes harder and harder to expand it in order to include more variables, by adding more states to the finite state machine. Not only is it conceptually more difficult to maintain a finite state machine with too many states, but additional slices are required as well. Moreover, such big finite state machines make it more difficult for the synthesis tools to produce the same circuit out of the VHDL code (Kelley, 2010).

We can reduce the size of the finite state machine and at the same time retain its functionality by converting the computational cores to tiny processors. This conversion is not as complicated as it may initially sound, since many of the units required for a simple processor, like the one described in (Patterson and Hennessy, 2005), are already included in the cores of our current design, such as the memory unit, the register file and the arithmetic unit. If the instruction set is kept small and simple, then the control logic of the processor, implemented as a finite state machine and functioning as an instruction decoder, may likewise be composed of just a few states, requiring less slices. Essentially, only arithmetic, load/store and branch instructions are necessary. A preliminary implementation of this design uses less than 15 instructions.

Of course, a small portion of the RAM memory should now be dedicated to storing the series of instructions which compute the new values. The organization of the memory units for each core should now look like that of fig. 5.1, assuming that the neural network is composed of neurons, synapses and (possibly) neuromodulators. It starts with a metadata section in which pointers to the other sections are stored. The next section is the one that stores the "programs" which the processor has to run and is followed be the parameters' section which, as implied by its name, holds the values for the various parameters, like the time constants, the reversal potential etc. The data for the network itself (what was previously the only section, fig. 3.3) are held in the last section, obviously taking up most of the RAM.

Explicitly storing the parameters in the RAM, instead of using VHDL configuration files, gives us the ability to view the implementation of the design, in terms of synthesizing, mapping and routing the circuit, as a process distinct from loading a network. Since everything is now stored in the RAMs, including the parameters, we can implement the design with as many cores as we wish and later load (and reload) in the RAMs whichever network we want, provided of course that we have a way of sending the new network to the embedded system (one way would be over the serial port). Having the ability to quickly change the network our FPGA is supposed to simulate might not seem as a significant improvement. If we consider though that the whole process of implementing a new design on the FPGA might very well take 20 or more minutes (more time-consuming for complex designs), then it becomes clear that such a feature could be very helpful (it is really frustrating to wait 20 minutes in order to change the value of just one parameter).

Although this new design has increased requirements, as far as the RAM blocks are con-

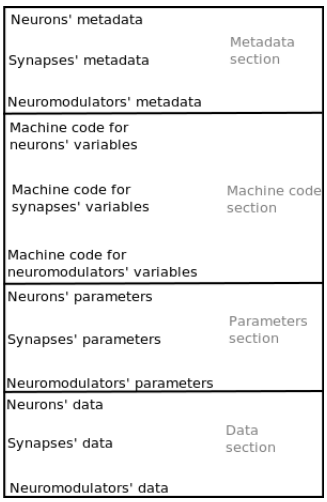


Figure 5.1: RAM organization for the future version of the hardware simulator

cerned, it is not expected that they will constitute a serious constraint or limit even further the size of the networks that the FPGA can host. As mentioned previously, a preliminary version has already been implemented. For the time being, it can simulate only neurons, since neither the machine code for synapses and neuromodulators nor the spikes' transmission mechanism have been included, but we can get a rough idea what the requirements for the final design are going to be. In order to simulate neurons, a RAM block needs to store three pointers in the metadata section, 48 "assembly" instructions and 10 parameters (only one neuron type). Considering that the data section requires (tens of) thousands of addresses for networks with hundreds or thousands of neurons, the overhead of the metadata, machine code and parameters sections is not significant.

With respect to the other hardware requirements of the design, besides those regarding memory, it was found that each core now needs almost half of the slices, compared to the old cores (fig. 3.4). Of course, this is an underestimation but the incorporation of the necessary modules for the synapses and the neuromodulators is not expected to have a very pronounced effect on this figure. It is essentially the spikes' transmission module which requires more slices whereas solving the equations for the synapses and neuromodulators requires only that the corresponding machine code be written. Measurements for the speed performance were not taken because the omission of synapses prevents us from making any reliable predictions. In fact, the computational load of the synapses exceeds by far that of the neurons. It could be argued that moving towards a more "software" based solution will probably make the design slower. However, smaller cores also means more cores and (usually) increased performance. Therefore, at this moment, we are not in a position to estimate the speed performance of the new design.

Some other techniques could also be of use in order to improve the speed performance. When someone watches the activity of a neural network, it is evident that there are certain time "gaps" in which the state of a neuron or a synapse remains the same, e.g. when a neuron is (or has returned) at its resting potential without any input currents or when a synapse is quiescent without any pre- or post-synaptic spikes. Sometimes, a neuron or a synapse might actually spend most of their time in such gaps of idleness. These gaps constitute wasted computational cycles for the simulator. An event-driven simulator could address this issue by ensuring that the computational path of each element is executed only when it is necessary.

The same approach could also be followed, at a higher abstraction level. As we have explained in 3.3.3, the simulator, when attached to the embedded system as a peripheral, works under direct control of the software drivers. The network is updated to its next state only when it receives an *update* pulse from Microblaze. This type of "synchronization" between the main processor and the "cluster" of neural processors allows for a complete control (useful when someone wants to check that the peripheral works as expected) but introduces a significant software delay (see again 3.3.3). We could eliminate this delay by letting the simulator run in a continuous mode, without the need for pulses from the main processor for each and every update cycle, checking at the beginning of every cycle whether there is new input or not. The software driver should then be responsible for starting (and stopping) the network and for sending new input values to it.

5.1.3 Robot experiments

With the improvements described in the previous section, we can be confident that our embedded system would be capable of simulating neural networks close in size (and hopefully complexity) to the olfactory system of flies. More complete models of the fly's olfactory pathway would open up new research paths and would allow us to examine more thoroughly some of the assumptions which we made in order to run our experiments.

One of them concerns the way sensory data are presented to the neural network (4.3). For the experiments presented in 4.2, we had to exert a considerable amount of control on the flow of sensory information to the robot's brain. By including another, sensory layer (OSNs) in our model, a more uninterrupted and biologically plausible sensory flow would become possible. Irrelevant as this might seem with respect to issues of memory and learning, we have reasons to believe (or at least assume, as a hypothesis) that the problem of perception might be more tightly linked to that of memory than usually acknowledged in many modelling studies. If plasticity can be observed even at the perceptual level, then it is very probable that this form of perceptual learning has a significant effect on the "higher" forms of learning. Conversely, we could justifiably expect the existence of top-down processing mechanisms from those "higher" memory structures (MBs) to perception (AL and OSNs). Therefore, the problem of how a

living organism (the fly in our case) manages to organize the stream of sensory data into stimuli might prove relevant to how they are actually stored in its brain.

In order to achieve this tighter integration of the robot within its environment, we would also need more advanced sensors, with the ability to give us a representation of a richer environment. Ideally, we would like to equip the robot with olfactory sensors with which to detect real odours. Since this is a technically very challenging task, we would have again to simulate olfaction, but in a more realistic manner this time. A simple way to achieve this is to use a camera but with a significant difference. Strange as it may sound, the camera should not be directed straight ahead but downwards. An odour could be represented by small dots on the floor, with the colour of the dots indicating different odours and their density indicating concentration. With certain image processing techniques, it is fairly easy to extract the dots from the background and determine their colour and number. Even the simple technique of template matching should suffice. The board which hosts the XC3SD3400A FPGA is equipped with a hardware interface for a camera and a small camera as well. In fact, this was the reason why it this FPGA kit was chosen. With such a setup, we could test the robot in arenas like the one depicted in fig. 5.2. It is worth noting that the arena is not simply an "external" factor in these experiments but has an important effect on the range of testable hypotheses. Not only could we manipulate the robot's brain (e.g. introduce plasticity in the ALs) but the environment as well. For example, we could test mixtures of odours or how stable the "representation" of an odour is under different concentrations, when the dots are closer or farther apart.

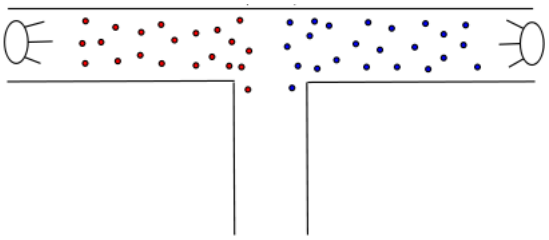


Figure 5.2: Diagram of an arena for running future robot experiments

In order to study the effects of top-down processing mechanisms (if there are any), we could run a more ambitious set of experiments. The robot would then be allowed to behave in a more exploratory mode, as in experiments with *Drosophila* larvae which have indicated that learned behaviour depends on outcome expectations (Gerber and Hendel, 2006). Such experiments would also give us the ability to investigate the relationship between memory and motor behaviour (see 4.3), the way motor circuits interact with the "higher" structures where motor commands initiate, the relationship between the UR and the CR and how they are affected by neuromodulators and a series of other issues concerning motor behaviour. Finally, it would be interesting to examine how the robot and its brain would behave if it could interact

with its environment, i.e. if it had the ability to actually change it according to its goals, possibly in cooperation with other robotic agents, instead of simply sensing it in a somewhat passive manner.

5.2 Conclusion

Throughout this thesis, we have attempted to cut across different levels of abstraction, from logic gates of digital circuits to embedded systems and to software simulators and from neurons and synapses to neural networks and to behaving agents, in order to catch a glimpse of the neural correlates of learning and memory in the insect brain. Hopefully, the reader should be by now convinced that this was not a futile attempt and that the whole process of actually building a biorobotic platform raises questions and points to directions which would be much more difficult to discern by following a more conventional modelling approach.

It should also be obvious that our approach entails a substantial amount of technical work. In fact, most of our time was devoted to setting up the whole system and connecting its various pieces together. However, we have managed to show that it is feasible to build autonomous robots with silicon brains, using FPGAs to simulate them. It is worth noting that the design we presented can simulate neural networks which are biologically plausible and does not make an effort to simplify the model in order to gain in efficiency, a crucial feature when the issues to be addressed pertain to neuroscience and not to engineering. We should bear in mind though that what is considered today as more relevant to memory and learning might be considered less important in some years. The field of neuroscience is rapidly evolving and it would be reckless to design an inflexible system. This was the reason behind the decision to move towards a more "programmable" architecture so that the new models and mechanisms which may be discovered can be incorporated into the system with a minimal effort.

Although the robot in our experiments carried a brain of just 81 neurons, the results from the more powerful FPGA board clearly show that it is possible to implement networks with hundreds of neurons and a slight modification could give us the ability to test brains with thousands or even tens of thousands of neurons in real-time, a figure which is close to the size of the fly's olfactory system. Moreover, the scaling behaviour of our design indicates that we can improve its speed performance, if need be, by simply adding more cores, provided that there are available hardware resources. We can therefore see that the technology of FPGAs, hard as it may be to master and comprehend in depth, can become a very useful tool for neuroroboticists in the future, especially if we consider that it keeps advancing and producing devices with more and more slices. Although it is tempting to do so, we will refrain from providing an estimate of how our design would perform with the latest series of Xilinx FPGAs (Virtex-6) some of which are 10 (or even more) times more powerful than ours in terms of slices, RAM

blocks and DSP elements. Even these FPGAs have their limits but we could imagine a system with multiple FPGA boards, each being responsible for simulating a specific subsystem of the nervous system (visual, auditory, olfactory etc) and all of them working together to simulate the whole nervous system. However, when we scale the system beyond a certain point, it is expected that new issues may arise, e.g. the problem of communication among the cores would probably become a severe bottleneck.

It is also doubtful that models of such a scale would be of much use, as far as understanding the neural mechanisms behind certain aspects of behaviour is concerned. Our robot experiments have shown that we do not necessarily have to search for answers in models of increasing complexity and scale. Even minimal models, like the one used in this thesis, can help us gain significant insights, especially when we need to understand the links between neural mechanisms and behaviour. Our results indicate that the architecture of the insect brain has the "computational" power to solve even non-elemental learning tasks. We probably need to consider what top-down mechanisms might be involved, in order to make our models compatible with the behavioural data. Insect learning is much more than simple reflexes. With regard to the specific neural plasticity mechanisms that might underlie learning, our conclusion is that STDP remains to a significant extent problematic. Although its modified version can solve the distal reward problem in theoretical models, its applicability to more realistic situations remains doubtful. We propose that the hypothesis of STDP functioning as a "priming" mechanism should be examined.

If computational neuroscience is now in a position where it can afford enough computational power to model the brain (human, rat, insect or otherwise) in part or even in whole, then this might also be the time to take these silicon brains out of the "tube" and throw them into the world (nothing in the vein of gnosticism is implied here, at least not "consciously"). Of course, for every new collider that physicists build, neuroscientists have every right to build a new supercomputer to model yet even more massive and detailed brain models. Or we can remind ourselves of Canguilhem's remarks and then take a look again at the poem with which this thesis begins.

Appendix A

Appendix A: MATLAB source code for the GUI

Due to space limitations, the source code has not been included to the manuscript. It can be located at the CVS repository *cvs.inf.ed.ac.uk:2401/disk/cvs/sacratio*.

Appendix B

Appendix B: C source code for the Microblaze application

Due to space limitations, the source code has not been included to the manuscript. It can be located at the CVS repository *cvs.inf.ed.ac.uk:2401/disk/cvs/sacratio*.

Appendix C

Appendix C: VHDL source code for the neural networks hardware simulator

Due to space limitations, the source code has not been included to the manuscript. It can be located at the CVS repository *cvs.inf.ed.ac.uk:2401/disk/cvs/sacratio*.

Bibliography

Almassy, N., Edelman, G. M., and Sporns, O. (1998). Behavioral constraints in the development of neuronal properties: A cortical model embedded in a real-world device. *Cerebral Cortex*, 8(4):346–61.

Arleo, A., Smeraldi, F., and Gerstner, W. (2004). Cognitive navigation based on nonuniform gabor space sampling, unsupervised growing networks, and reinforcement learning. *IEEE Transactions on Neural Networks*, 15(3):639–652.

Banquet, J. P., Gaussier, P., Quoy, M., Revel, A., and Burnod, Y. (2005). A hierarchy of associations in Hippocampo-Cortical systems: Cognitive maps and navigation strategies. *Neural Computation*, 17(6):1339–1384.

Bazhenov, M., Stopfer, M., Rabinovich, M., Huerta, R., Abarbanel, H. D., Sejnowski, T. J., and Laurent, G. (2001). Model of transient oscillatory synchronization in the locust antennal lobe. *Neuron*, 30(2):553–67.

Bazhenov, M., Stopfer, M., Sejnowski, T. J., and Laurent, G. (2005). Fast odor learning improves reliability of odor responses in the locust antennal lobe. *Neuron*, 46(3):483–492.

Beer, R. (2008). *The dynamics of brain-body-environment systems: A status report In Handbook of Cognitive Science: An Embodied Approach. P. Calvo and A. Gomila (Eds). Elsevier*.

Canguilhem, G. (1952). *Knowledge of Life*. Fordham UP.

Canguilhem, G. (1968). *Le normal et le pathologique, augmente de Nouvelles reflexions concernant le normal et le pathologique*.

Cassenaer, S. and Laurent, G. (2007). Hebbian STDP in mushroom bodies facilitates the synchronous flow of olfactory information in locusts. *Nature*, 448:709–714.

Cetina, K. K. (1999). *Epistemic Cultures. How the sciences make knowledge.* Harvard University Press.

Churchland, P. S., Ramachandran, V. S., and Sejnowski, T. J. (1994). A critique of pure vision. In Large-scale neuronal theories of the brain. C. Koch and J. L. Davis (Eds). The MIT press.

Cuperlier, N., Quoy, M., and Gaussier, P. (2007). Neurobiologically inspired mobile robot navigation and planning. *Frontiers in Neurorobotics*, 1.

Daly, K. C., Christensen, T. A., Lei, H., Smith, B. H., and Hildebrand, J. G. (2004). Learning modulates the ensemble representations for odors in primary olfactory networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(28):10476–10481.

Damper, R. I., French, R. L. B., and Scutt, T. W. (2000). ARBIB: An autonomous robot based on inspirations from biology. *Robotics and Autonomous Systems*, 31(4):247–274.

Dan, Y. and Poo, M. (2004). Spike timing-dependent plasticity of neural circuits. *Neuron*, 44(1):23–30.

DasGupta, S. and Waddell, S. (2008). Learned odor discrimination in drosophila without combinatorial odor maps in the antennal lobe. *Current Biology*, 18(21):1668–1674.

Davis, R. L. (2004). Olfactory learning. *Neuron*, 44(1):31–48.

Davis, R. L. (2005). Olfactory memory formation in drosophila: from molecular to systems neuroscience. *Annual Review of Neuroscience*, 28:275–302.

Day, R. (2001). *The Modern Invention of Information: Discourse, History, and Power*. Southern Illinois University Press.

Dayan, P. and Abbott, L. F. (2001). *THEORETICAL NEUROSCIENCE Computational and Mathematical Modelling of Neural Systems*. The MIT press.

Deisig, N., Lachnit, H., Giurfa, M., and Hellstern, F. (2001). Configural olfactory learning in honeybees: Negative and positive patterning discrimination. *Learning & Memory*, 8(2):70–78.

Destexhe, A., Mainen, Z. F., and Sejnowski, T. J. (1994). Synthesis of models for excitable membranes, synaptic transmission and neuromodulation using a common kinetic formalism. *Journal of Computational Neuroscience*, 1(3):195–230.

Dobritsa, A. A., van der Goes van Naters, W., Warr, C. G., Steinbrecht, R. A., and Carlson, J. R. (2003). Integrating the molecular and cellular basis of odor coding in the drosophila antenna. *Neuron*, 37(5):827–41.

Doya, K. and Uchibe, E. (2005). The cyber rodent project: Exploration of adaptive mechanisms for Self-Preservation and Self-Reproduction. *Adaptive Behavior*, 13(2):149–160.

Dreyfus, H. L. (2002). Intelligence without representation - Merleau-Ponty's critique of mental representation the relevance of phenomenology to scientific explanation. *Phenomenology and the Cognitive Sciences*, 1(4):367–383.

Dubnau, J., Grady, L., Kitamoto, T., and Tully, T. (2001). Disruption of neurotransmission in drosophila mushroom body blocks retrieval but not acquisition of memory. *Nature*, 411(6836):476–480.

Edelman, G. M. (2007). Learning in and from brain-based devices. *Science*, 318(5853):1103–1105.

Engel, A. K., Fries, P., and Singer, W. (2001). Dynamic predictions: oscillations and synchrony in top-down processing. *Nature Reviews. Neuroscience*, 2(10):704–716.

Faber, T., Joerges, J., and Menzel, R. (1999). Associative learning modifies neural representations of odors in the insect brain. *Nature Neuroscience*, 2(1):74–78.

Fahrbach, S. E. (2006). Structure of the mushroom bodies of the insect brain. *Annual Review of Entomology*, 51:209–32.

Fleischer, J. G., Gally, J. A., Edelman, G. M., and Krichmar, J. L. (2007). Retrospective and prospective responses arising in a modeled hippocampus during maze navigation by a brain-based device. *Proceedings of the National Academy of Sciences*, 104(9):3556–3561.

Floreano, D. and Keller, L. (2010). Evolution of adaptive behaviour in robots by means of darwinian selection. *PLoS Biology*, 8(1):1–8.

Franceschini, N. (2008). Towards automatic visual guidance of aerospace vehicles: from insects to robots. *Acta Futura*, 4:12–28.

Freeman, W. (1975). *Mass Action in the Nervous System*. Academic Press.

Freeman, W. J. (2007). Intentionality. *Scholarpedia*, 2(2):1337.

Freeman, W. J. (2008). Freeman k-set. *Scholarpedia*, 3(2):3238.

Gao, Q., Yuan, B., and Chess, A. (2000). Convergent projections of drosophila olfactory neurons to specific glomeruli in the antennal lobe. *Nature Neuroscience*, 3(8):780–5.

Gerber, B. and Hendel, T. (2006). Outcome expectations drive learned behaviour in larval drosophila. *Proceedings. Biological Sciences / The Royal Society*, 273(1604):2965–2968.

Ghani, A., McGinnity, T., Maguire, L., and Harkin, J. (2006). Area efficient architecture for large scale implementation of biologically plausible spiking neural networks on reconfigurable hardware. In *International Conference on Field Programmable Logic and Applications, 2006. FPL '06.*, pages 1–2.

Gibson, J. J. (1979). *The Ecological Approach to Visual Perception*. Boston: Houghton Mifflin.

Gleeson, P., Crook, S., Cannon, R. C., Hines, M. L., Billings, G. O., Farinella, M., Morse, T. M., Davison, A. P., Ray, S., Bhalla, U. S., Barnes, S. R., Dimitrova, Y. D., and Silver, R. A. (2010). Neuroml: A language for describing data driven models of neurons and networks with a high degree of biological detail. *PLoS Computational Biology*, 6(6).

Gouwens, N. W. and Wilson, R. I. (2009). Signal propagation in drosophila central neurons. *The Journal of Neuroscience*, 29(19):6239–6249.

Graas, E. L., Brown, E. A., and Lee, R. H. (2004). An FPGA-based approach to high-speed simulation of conductance-based neuron models. *Neuroinformatics*, 2(4):417–36.

Grunewald, B. (1999). Morphology of feedback neurons in the mushroom body of the honeybee, apis mellifera. *The Journal of Comparative Neurology*, 404(1):114–26.

Hallem, E. A. and Carlson, J. R. (2004). The odor coding system of drosophila. *Trends in Genetics*, 20(9):453–9.

Hallem, E. A., Ho, M. G., and Carlson, J. R. (2004). The molecular basis of odor coding in the drosophila antenna. *Cell*, 117(7):965–79.

Hammer, M. (1993). An identified neuron mediates the unconditioned stimulus in associative olfactory learning in honeybees. *Nature*, 366(6450):59–63.

Hammer, M. and Menzel, R. (1998). Multiple sites of associative odor learning as revealed by local brain microinjections of octopamine in honeybees. *Learning & Memory*, 5(1-2):146–56.

Heisenberg, M. (1998). What do the mushroom bodies do for the insect brain? an introduction. *Learning & Memory*, 5(1-2):1–10.

Heisenberg, M. (2003). Mushroom body memoir: from maps to models. *Nature Reviews. Neuroscience*, 4(4):266–75.

Hildebrand, J. G. and Shepherd, G. M. (1997). Mechanisms of olfactory discrimination: converging evidence for common principles across phyla. *Annual Review of Neuroscience*, 20:595–631.

Holzkamp, K. (1995). *Lernen: subjekwissentschaftliche Grundlegung*. Campus Verlag.

Homberg, U., Christensen, T. A., and Hildebrand, J. G. (1989). Structure and function of the deutocerebrum in insects. *Annual Review of Entomology*, 34:477–501.

Horchler, A. D., Reeve, R. E., Webb, B., and Quinn, R. D. (2004). Robot phonotaxis in the wild: a biologically inspired approach to outdoor sound localization. *Advanced Robotics*, 18(8):801–816.

Huerta, R. and Nowotny, T. (2009). Fast and robust learning by reinforcement signals: Explorations in the insect brain. *Neural Computation*, 21(0):1–29.

Ijspeert, A. J. (2008). Central pattern generators for locomotion control in animals and robots: a review. *Neural Networks: The Official Journal of the International Neural Network Society*, 21(4):642–653. PMID: 18555958.

Ito, I., Ong, R. C., Raman, B., and Stopfer, M. (2008). Sparse odor representation and olfactory learning. *Nature Neuroscience*, 11:1177–1184.

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, 14(6):1569–1572.

Izhikevich, E. M. (2004). Which model to use for cortical spiking neurons? *IEEE Transactions on Neural Networks*, 15(5):1063–70.

Izhikevich, E. M. (2007). Solving the distal reward problem through linkage of STDP and dopamine signaling. *Cerebral Cortex*, 17:2443–2452.

Jarvilehto, T. (1999). The theory of the organism-environment system: III. role of efferent influences on receptors in the formation of knowledge. *Integrative Physiological and Behavioral Science*, 34(2):90–100.

Jones, W. D., Nguyen, T. T., Kloss, B., Lee, K. J., and Vosshall, L. B. (2005). Functional conservation of an insect odorant receptor gene across 250 million years of evolution. *Current Biology*, 15(4):R119–21.

K-Team (1999). *KOALA user manual*. K-Team.

Keene, A. C., Krashes, M. J., Leung, B., Bernard, J. A., and Waddell, S. (2006). Drosophila dorsal paired medial neurons provide a general mechanism for memory consolidation. *Current Biology*, 16(15):1524–1530.

Keene, A. C., Stratmann, M., Keller, A., Perrat, P. N., Vosshall, L. B., and Waddell, S. (2004). Diverse odor-conditioned memories require uniquely timed dorsal paired medial neuron output. *Neuron*, 44(3):521–533.

Keene, A. C. and Waddell, S. (2007). Drosophila olfactory memory: single genes to complex neural circuits. *Nature Reviews. Neuroscience*, 8(5):341–54.

Kelley, K. (2010). *WP361. Xilinx white paper: Maintaining repeatable results.* Xilinx.

Krashes, M. J., Keene, A. C., Leung, B., Armstrong, J. D., and Waddell, S. (2007). Sequential use of mushroom body neuron subsets during drosophila odor memory processing. *Neuron*, 53(1):103–115.

Krichmar, J. (2008). Neurorobotics. *Scholarpedia*, 3(3):1365.

Krichmar, J. L. and Edelman, G. M. (2002). Machine psychology: Autonomous behavior, perceptual categorization and conditioning in a brain-based device. *Cereb. Cortex*, 12(8):818–830.

Krichmar, J. L., Seth, A. K., Nitz, D. A., Fleischer, J. G., and Edelman, G. M. (2005). Spatial navigation and causal analysis in a brain-based device modeling cortical-hippocampal interactions. *Neuroinformatics*, 3(3):197–221.

Laissue, P. P., Reiter, C., Hiesinger, P. R., Halter, S., Fischbach, K. F., and Stocker, R. F. (1999). Three-dimensional reconstruction of the antennal lobe in drosophila melanogaster. *The Journal of Comparative Neurology*, 405(4):543–52.

Laurent, G. (2002). Olfactory network dynamics and the coding of multidimensional signals. *Nature Reviews. Neuroscience*, 3(11):884–95.

Linster, C., Sachse, S., and Galizia, C. G. (2005). Computational modeling suggests that response properties rather than spatial position determine connectivity between olfactory glomeruli. *Journal of Neurophysiology*, 93(6):3410–3417.

Liu, L., Wolf, R., Ernst, R., and Heisenberg, M. (1999). Context generalization in drosophila visual learning requires the mushroom bodies. *Nature*, 400(6746):753–756.

Mackintosh, N. J. (1975). A theory of attention: Variations in the associability of stimuli with reinforcement. *Psychological Review*, 82:276–198.

Maguire, L., McGinnity, T., Glackin, B., Ghani, A., Belatreche, A., and Harkin, J. (2007). Challenges for large-scale implementations of spiking neural networks on FPGAs. *Neurocomputing*, 71(1-3):13–29.

Marcuse, H. (1964). *One-Dimensional Man.* Routledge.

Margulies, C., Tully, T., and Dubnau, J. (2005). Deconstructing memory in drosophila. *Current Biology*, 15(17):R700–13.

Marin, E. C., Jefferis, G. S. X. E., Komiyama, T., Zhu, H., and Luo, L. (2002). Representation of the glomerular olfactory map in the drosophila brain. *Cell*, 109(2):243–55.

Markram, H. (2006). The blue brain project. *Nature Reviews. Neuroscience*, 7(2):153–160. PMID: 16429124.

Maxfield, C. (2004). *The Design Warrior's Guide to FPGAs*. Newnes.

McGuire, S. E., Le, P. T., and Davis, R. L. (2001). The role of drosophila mushroom body signaling in olfactory memory. *Science*, 293(5533):1330–1333.

McGuire, S. E., Le, P. T., Osborn, A. J., Matsumoto, K., and Davis, R. L. (2003). Spatiotemporal rescue of memory dysfunction in drosophila. *Science*, 302:1765–1768.

Milford, M. J., Prasser, D., and Wyeth, G. (2004). RATSLAM: a hippocampal model for simultaneous localization and mapping. In *IEEE International Conference on Robotics and Automation, 2004.*, pages 403–408.

Muezzinoglu, M. K., Huerta, R., Abarbanel, H. D. I., Ryan, M. A., and Rabinovich, M. I. (2009). Chemosensor-driven artificial antennal lobe transient dynamics enable fast recognition and working memory. *Neural Computation*, 21(4):1018–1037.

Nowotny, T., Huerta, R., Abarbanel, H. D. I., and Rabinovich, M. I. (2005). Self-organization in the olfactory system: one shot odor recognition in insects. *Biological Cybernetics*, 93(6):436–446.

Nunez, R. and Freeman, W. J. (1999). Restoring to cognition the forgotten primacy of action, intention and emotion. *Journal of Consciousness Studies*, 6(11-12):ix–xix.

Ortigosa, E., Caas, A., Ros, E., Ortigosa, P., Mota, S., and Daz, J. (2006). Hardware description of multi-layer perceptrons with different abstraction levels. *Microprocessors and Microsystems*, 30(7):435 – 444.

Pascual, A. and Preat, T. (2001). Localization of long-term memory within the drosophila mushroom bodies. *Science*, 294:1115–1117.

Patterson, D. A. and Hennessy, J. L. (2005). *Computer Organization and Design. The Hardware/Software Interface.* Morgan Kaufmann, 3rd edition.

Pearce, J. M. and Bouton, M. E. (2001). Theories of associative learning in animals. *Annual Review of Psychology*, 52:111–139.

Pearce, J. M. and Hall, G. (1980). A model for pavlovian learning: variations in the effectiveness of conditioned but not of unconditioned stimuli. *Psychological Review*, 87(6):532–52.

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.

Perez-Orive, J., Bazhenov, M., and Laurent, G. (2004). Intrinsic and circuit properties favor coincidence detection for decoding oscillatory input. *The Journal of Neuroscience*, 24(26):6037–6047.

Perez-Orive, J., Mazor, O., Turner, G. C., Cassenaer, S., Wilson, R. I., and Laurent, G. (2002). Oscillations and sparsening of odor representations in the mushroom body. *Science*, 297(5580):359–65.

Pfeifer, R., Lungarella, M., and Iida, F. (2007). Self-organization, embodiment, and biologically inspired robotics. *Science*, 318(5853):1088–1093.

Prescott, T. J., Gonzlez, F. M. M., Gurney, K., Humphries, M. D., and Redgrave, P. (2006). A robot model of the basal ganglia: behavior and intrinsic processing. *Neural Networks*, 19(1):31–61.

Rajashekhar, K. P. and Singh, R. N. (1994). Neuroarchitecture of the tritocerebrum of drosophila melanogaster. *The Journal of Comparative Neurology*, 349(4):633–45.

Rescorla, R. A. and Wagner, A. R. (1972). A theory of pavlovian conditioning: variations in the effectiveness of reinforcement and non-reinforcement. In A.H. Black and W.F. Prokasy (Eds.), Classical Conditioning II: Theory and Research.

Riemensperger, T., Vller, T., Stock, P., Buchner, E., and Fiala, A. (2005). Punishment prediction by dopaminergic neurons in drosophila. *Current Biology*, 15(21):1953–1960.

Roberts, P. D. and Bell, C. C. (2002). Spike timing dependent plasticity in biological systems. *Biological Cybernetics*, 87:392–403.

Roggen, D., Hofmann, S., Thoma, Y., and Floreano, D. (2003). Hardware spiking neural network with run-time reconfigurable connectivity in an autonomous robot. In *NASA/DoD Conference on Evolvable Hardware, 2003. Proceedings.*, pages 189–198.

Ros, E., Ortigosa, E. M., Ags, R., Carrillo, R., and Arnold, M. (2006). Real-time computing platform for spiking neurons (RT-spike). *IEEE Transactions on Neural Networks*, 17(4):1050–1063.

Schmajuk, N. A. (2008). Computational models of classical conditioning. *Scholarpedia*, 3(3):1664.

Schroll, C., Riemensperger, T., Bucher, D., Ehmer, J., Vller, T., Erbguth, K., Gerber, B., Hendel, T., Nagel, G., Buchner, E., and Fiala, A. (2006). Light-induced activation of distinct

modulatory neurons triggers appetitive or aversive learning in drosophila larvae. *Current Biology*, 16(17):1741–1747.

Schubert, M., Lachnit, H., Francucci, S., and Giurfa, M. (2002). Nonelemental visual learning in honeybees. *Animal Behaviour*, 64(2):175–184.

Schwaerzel, M., Monastirioti, M., Scholz, H., Friggi-Grelin, F., Birman, S., and Heisenberg, M. (2003). Dopamine and octopamine differentiate between aversive and appetitive olfactory memories in drosophila. *The Journal of Neuroscience*, 23(33):10495–502.

Seth, A., Sporns, O., and Krichmar, J. (2005). Neurorobotic models in neuroscience and neuroinformatics. *Neuroinformatics*, 3(3):167–170.

Shanbhag, S. R., Mller, B., and Steinbrecht, R. A. (1999). Atlas of olfactory organs of drosophila melanogaster 1.types, external organization, innervation and distribution of olfactory sensilla. *International Journal of Insect Morphology & Embryology*, 28:277–97.

Shanbhag, S. R., Mller, B., and Steinbrecht, R. A. (2000). Atlas of olfactory organs of drosophila melanogaster 2. internal organization and cellular architecture of olfactory sensilla. *Arthropod Structure & Development*, 29:211–29.

Shang, Y., Claridge-Chang, A., Sjulson, L., Pypaert, M., and Miesenbck, G. (2007). Excitatory local circuits and their implications for olfactory processing in the fly antennal lobe. *Cell*, 128(3):601–612.

Sivan, E. and Kopell, N. (2004). Mechanism and circuitry for clustering and fine discrimination of odors in insects. *Proceedings of the National Academy of Sciences of the United States of America*, 101(51):17861–17866.

Smith, D., Wessnitzer, J., and Webb, B. (2008). A model of associative learning in the mushroom body. *Biological Cybernetics*, 99(2):89–103.

Song, S., Miller, K. D., and Abbott, L. F. (2000a). Competitive hebbian learning through spike timing-dependent synaptic plasticity. *Nature Neuroscience*, 3:919–926.

Song, S., Miller, K. D., and Abbott, L. F. (2000b). Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, 3(9):919–926.

Sporns, O. and Alexander, W. H. (2002). Neuromodulation and plasticity in an autonomous robot. *Neural Networks*, 15(4-6):761–774.

Stocker, R. F. (1994). The organization of the chemosensory system in drosophila melanogaster: a review. *Cell and Tissue Research*, 275(1):3–26.

Stocker, R. F., Lienhard, M. C., Borst, A., and Fischbach, K. F. (1990). Neuronal architecture of the antennal lobe in drosophila melanogaster. *Cell and Tissue Research*, 262(1):9–34.

Strausfeld, N. J., Hansen, L., Li, Y., Gomez, R. S., and Ito, K. (1998). Evolution, discovery, and interpretations of arthropod mushroom bodies. *Learning & Memory*, 5(1-2):11–37.

Tanaka, N. K., Tanimoto, H., and Ito, K. (2008). Neuronal assemblies of the drosophila mushroom body. *The Journal of Comparative Neurology*, 508(5):711–755.

Tang, S. and Guo, A. (2001). Choice behavior of drosophila facing contradictory visual cues. *Science*, 294(5546):1543–1547.

Tully, T., Preat, T., Boynton, S. C., and Vecchio, M. D. (1994). Genetic dissection of consolidated memory in drosophila. *Cell*, 79(1):35–47.

Tully, T. and Quinn, W. G. (1985). Classical conditioning and retention in normal and mutant drosophila melanogaster. *Journal of Comparative Physiology. A, Sensory, Neural, and Behavioral Physiology*, 157(2):263–77.

Turner, G. C., Bazhenov, M., and Laurent, G. (2008). Olfactory representations by drosophila mushroom body neurons. *Journal of Neurophysiology*, 99(2):734–46.

Upegui, A., Andres, C., and Sanchez, E. (2005). An FPGA platform for on-line topology exploration of spiking neural networks. *Microprocessors and Microsystems*, 29(5):211–223.

Vosshall, L. B., Wong, A. M., and Axel, R. (2000). An olfactory sensory map in the fly brain. *Cell*, 102(2):147–59.

Waddell, S., Armstrong, J. D., Kitamoto, T., Kaiser, K., and Quinn, W. G. (2000). The amnesiac gene product is expressed in two neurons in the drosophila brain that are critical for memory. *Cell*, 103(5):805–813.

Wagner, A. (1981). SOP: A model of automatic memory processing in animal behavior. In N.E. Spear and R.R. Miller (Eds.), Information processing in animals: Memory mechanisms (pp. 5-47). Hillsdale, NJ: Erlbaum.

Wang, Y., Guo, H., Pologruto, T. A., Hannan, F., Hakker, I., Svoboda, K., and Zhong, Y. (2004). Stereotyped odor-evoked activity in the mushroom body of drosophila revealed by green fluorescent protein-based ca2+ imaging. *The Journal of Neuroscience*, 24(29):6507–6514.

Webb, B. (2001). Can robots make good models of biological behaviour? *The Behavioral and Brain Sciences*, 24(6):1033–1050.

Webb, B. (2002). Robots in invertebrate neuroscience. *Nature*, 417(6886):359–363.

Weinstein, R. K. and Lee, R. H. (2006). Architectures for high-performance FPGA implementations of neural models. *Journal of Neural Engineering*, 3(1):21–34.

Weinstein, R. K., Reid, M. S., and Lee, R. H. (2007). Methodology and design flow for assisted neural-model implementations in FPGAs. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 15(1):83–93.

Weng, J., McClelland, J., Pentland, A., Sporns, O., Stockman, I., Sur, M., and Thelen, E. (2001). ARTIFICIAL INTELLIGENCE: autonomous mental development by robots and animals. *Science*, 291(5504):599–600.

Wessnitzer, J., Webb, B., and Smith, D. (2007). A model of non-elemental associative learning in the mushroom body neuropil of the insect brain. *Proceedings of the International Conference on Adaptive and Natural Computing Algorithms*, 4431/2007:488–497.

Wilson, R. I. and Laurent, G. (2005). Role of GABAergic inhibition in shaping odor-evoked spatiotemporal patterns in the drosophila antennal lobe. *The Journal of Neuroscience*, 25(40):9069–79.

Wong, A. M., Wang, J. W., and Axel, R. (2002). Spatial representation of the glomerular map in the drosophila protocerebrum. *Cell*, 109(2):229–41.

Wustenberg, D. G., Boytcheva, M., Grunewald, B., Byrne, J. H., Menzel, R., and Baxter, D. A. (2004). Current- and voltage-clamp recordings and computer simulations of kenyon cells in the honeybee. *Journal of Neurophysiology*, 92(4):2589–2603.

Xilinx (2007). *Xilinx DS529 Spartan-3A FPGA Family Data Sheet*. Xilinx.

Xilinx (2008). *Xilinx DS610 Spartan-3A DSP FPGA Family Data Sheet*. Xilinx.

Xilinx (2009). *Microblaze processor reference guide*. Xilinx.

Young, J. M., Wessnitzer, J., Armstrong, J. D., and Webb, B. (2010). Dissecting complex learning in drosophila: What are the limits of the fly brain? (submitted).

Yu, D., Keene, A. C., Srivatsan, A., Waddell, S., and Davis, R. L. (2005). Drosophila DPM neurons form a delayed and branch-specific memory trace after olfactory classical conditioning. *Cell*, 123(5):945–957.

Yu, D., Ponomarev, A., and Davis, R. L. (2004). Altered representation of the spatial code for odors after olfactory classical conditioning; memory trace formation by synaptic recruitment. *Neuron*, 42(3):437–49.

Zars, T., Fischer, M., Schulz, R., and Heisenberg, M. (2000). Localization of a short-term memory in drosophila. *Science*, 288(5466):672–5.

Zhang, J., Lau, P., and Bi, G. (2009). Gain in sensitivity and loss in temporal contrast of STDP by dopaminergic modulation at hippocampal synapses. *Proceedings of the National Academy of Sciences of the United States of America*, 106(31):13028–13033.

Zhang, W. and Linden, D. J. (2003). The other side of the engram: experience-driven changes in neuronal intrinsic excitability. *Nature Reviews. Neuroscience*, 4(11):885–900.

Zizek, S. (2009). *First as tragedy, then as farce*. Verso.