Novel Neuromorphic Hardware Inspired by the Olfactory Pathway Model of the Drosophila

Leveraging bio-plausible computational primitives in digital circuits for spatio-temporal processing

Thesis Report

Lyana Usa

Delft University of Technology

TU Delft

Novel Neuromorphic Hardware Inspired by the Olfactory Pathway Model of the Drosophila

Leveraging bio-plausible computational primitives in digital circuits for spatio-temporal processing

Thesis report

by

Lyana Usa

to obtain the degree of Master of Science
at the Delft University of Technology
to be defended publicly on April 29, 2024 at 15:00.

Thesis committee:	
Chair:	Prof. Dr. K. Makinwa
Supervisor:	Dr. C. Frenkel
External examiners:	Dr. G.C.H.E. de Croon
	Prof. Dr. M. P. Nawrot
Place:	Faculty of EEMCS, Delft
Student number:	4650352

TU Delft

Abstract

Olfactory learning in *Drosophila* larvae exemplifies efficient neural processing in a small-scale network with minimal power consumption. This system enables larvae to anticipate important outcomes based on new and familiar odor stimuli, a process crucial for survival and adaptation. Central to this learning mechanism is the olfactory pathway model, which embodies the principles of synaptic plasticity and associative learning through prediction error coding mediated by specific neuromodulating neurons in the mushroom body, like dopaminergic neurons. There is a pressing need to develop novel computational frameworks that capture the spatio-temporal processes while remaining compatible with the constraints of small-scale neural networks. These frameworks should draw inspiration from the biophysical properties of neurons within the olfactory pathway model, enabling accurate emulation of neural dynamics and efficient learning processes using spiking neural networks. This thesis proposes a framework based on a phenomenological conductance-based leaky integrate-and-fire (COBALIF) neuron model, inspired by the olfactory pathway model of *Drosophila* larvae. By first prototyping the spiking neural network in Intel's Lava Python-based framework, we validated the design on a neuron and system level for a neuromorphic hardware implementation. This was the foundation of a programmable, neuromorphic FPGA architecture capable of adaptive optimization, employed on a Zynq 7000 SoC FPGA. By implementing this architecture in a single-precision floating-point format, we model the real-time neural dynamics of the COBALIF neuron in one-tenth of a millisecond precision. Moreover, our FPGA implementation serves as a feasible prototype for deploying such biologically inspired neurons and their spatio-temporal dependencies in digital design, paving the way for scaling up to small-scale networks.

Acknowledgement

"Life is peak and valleys. Some people explain it as good times and bad times. But I think it's: Learning, Mastership, Learning, Mastership, okay? "

Lauryn Hill

I am immensely grateful to my family and friends for their support throughout this journey at university. To my parents, we have come a long way from Aceh, starting from the bottom, and now look at us. I would not have been able to do this without your unconditional love and support. I am eternally grateful for the opportunities you have given me, and for fuelling me with the strength and courage from our roots and culture. I am indebted to my brothers, Atjeh and Daniel, for their love, encouragement, and sibling quarrels. Thank you for keeping me sane and grounded. I am deeply thankful for my high school friends (Thirza, Josephine, Kirti, Iris, Nica, Kimberley where you at?!), who have stuck with me since we were 11. Your support has been my anchor in times of uncertainty in the past years. Our story is still being written. I am deeply thankful for my old and new family members. I am thankful for my home girl, Sabriena, whom I have known since kindergarten and who has always been the greatest cheerleader. Cyd, I am so happy that we are now family. Thank you for bringing sunshine to our tiny home and family. My friends from the internship in Paris, I am grateful for the memories we have made and your support from a distance. To my university friends (you know who you are), we have had the craziest experiences, which I will cherish forever. Besides that, I am grateful for our memories, especially our trips, and our breaks from studying and pulling those all-nighters. Matthew, I am deeply grateful for your unconditioned love and patience. You and I know that it has not been easy, but I am eternally grateful for your understanding and embracing me for who I am. The next step in my life has come and I am grateful to take you all with me. I thank my supervisor, Charlotte, for the guidance and constructive feedback. I would also like to sincerely thank Martin and Anna for their invaluable insights and support throughout this project. Thank you for inviting me to your faculty in Cologne. Lastly, I would like to acknowledge the support provided by Intel Neuromorphic Research Community Lab during your boot camp in Munich, and for providing the necessary resources essential for the completion of the thesis.

Contents

List of Figures

List of Tables

1 Introduction

Studies on the insect nervous system have significantly contributed to our understanding of the insect's brain, physiology, and neurobiology [1]. The insect brain can process, learn, and adapt to sensory input with a minimal power expenditure thanks to a small brain. For example, a honey bee's brain has a volume of less than 1 mm^3 consisting of about one million neurons with a brain power consumption of approximately 10 µW [2, 3]. The combination of low power consumption, adaptability, and online learning makes the olfactory pathway model of the *Drosophila* particularly appealing to emerging neuromorphic computing and engineering. Conventional computing systems are subjected to the end of Moore's Law and Dennard scaling, which calls for a shift in the research landscape. Neuromorphic computing explores new technologies to support the increasing demand for computing performance. In the late 1980s, the term *neuromorphic computing* was coined by the American scientist and engineer, Carver Mead [4, 5]. He showed how the bio-physical processes can be emulated with silicon technology [4, 5, 6]. The advent of neuromorphic computing systems contributed to the development of unconventional computing systems, employing principles observed in animal nervous systems [7, 8, 9]. Edge computing has resulted in many applications in medicine, smart sensing, and aerospace [10]. These applications are governed by power and size restrictions and ideally have a low memory footprint for potentially highly exhaustive and variable input data [10]. This is where we distinguish offline cloud-based deep learning from ”learning at the edge”. In this case, the computations of data occur close to where the data is collected, which is ruled by its power consumption that is not prioritized in the design space of deep learning. Similar constraints are found in the neural systems of insects. Despite their low neural footprint, they have excellent learning abilities and can perform complex, multi-sensory tasks [11].

Olfactory pathway model of the *Drosophila*
Neurons are the principal units for performing computations and establishing communication of neural information in the brains and nervous systems of (in)vertebrates. Specifically, the brain of the *Drosophila (Melanogaster)* - commonly referred to as the fruit fly - is a well-studied model. Their brain is 331,565$\times$ smaller than the human's brain [13, 14]. Many studies have been conducted to understand the fascinating relation between complex, sensory processing and the reduced neural network sizes of the insect brain. The small, olfactory pathway model of the *Drosophila* has received much attention due to its ability to perform chemo-sensory processing to anticipate and learn behavioral

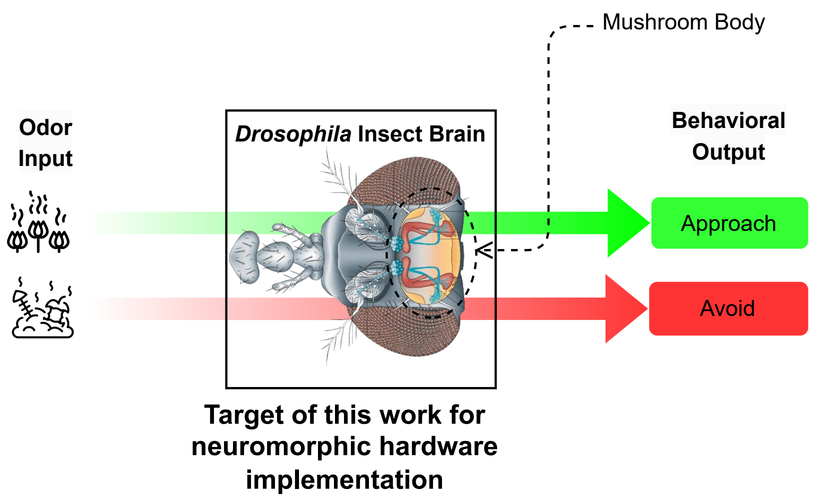

Mushroom Body
Odor Input
Drosophila Insect Brain
Behavioral Output
Approach
Avoid
Target of this work for neuromorphic hardware implementation

Figure 1.1: Schematic drawing of the processing pipeline of the *Drosophila's* olfaction model. At the input of the olfactory pathway, the odor's chemical information is transduced and encoded into electrical signals. In this example, the inputs are scents from flowers and litter. The processing of the pathway aims to generate outputs to control the behavioral and motor responses, which happens at the higher-processing center, i.e. the mushroom body of the insect brain. In the mushroom body, the dense, sensory information is sparsely encoded. The transformed, sensory information then proceeds in the mushroom body where the processing and learning of association happens. Finally, the mushroom body generates behavioral outputs - e.g. to avoid or to approach - depending on the learned response to a specific odor. This pathway model is the target of this thesis for neuromorphic hardware implementation. The cartoon of the *Drosophila* head is extracted from [12].

outputs, e.g. to approach or avoid an odor source based on its learned response. Its olfactory anatomy has been verified through *in-vivo* experiments [15, 16], which revealed key biological principles for olfactory learning [12]. The olfactory processing center is located in the *mushroom body* inside the insect brain, which is indicated in Figure 1.1. The mushroom body enables the construction of biological, associative memory for odors and olfactory learning. Learned and memorized experiences are used to anticipate the correct behavioral response to familiar environments and new contexts [17]. Figure 1.1 shows a top-level illustration of how the sense of smell is processed in the *Drosophila*'s olfactory pathway. The olfactory input to the olfactory pathway is detected and received by the antennas of the insect. The input information consists of chemical cues, which is transduced to electrical signals before entering the mushroom body. Then, in the mushroom body, the dense, odorant information is transformed into a sparse code [18]. This sparseness is employed in space (spatial) and in time (temporal), resulting in a spatio-temporal transformation. The odorant information in the mushroom body is represented by a small subset of all neurons (population or spatial sparseness) [19]. In other words, the overall response reduces for at least ten times for a given odor spatially [20, 21]. Furthermore, each activated neuron elicits only few action potentials, accounting for temporal sparseness [19]. The background firing rate in the mushroom body is very low with a mean firing rate of maximally 0.4 Hz across the entire population [22, 18]. During odor presentations, this rate increases to 5 and 20 Hz [23].

In the mushroom body, the sparse code takes a spatially and temporally sparse spatio-temporal format that is proven to be crucial in the efficient classification of odors on a tiny neural area [19, 24, 25], supporting efficient encoding and dynamic processing of olfactory data on its low neural footprint [26, 27, 28]. The concept of this olfactory encoding is depicted in Figure 1.2. In Figure 1.2, at the front-end of the mushroom body, the neurons form many links with other neurons, representing the dense representation of the odor input. Lastly, at the end of the pathway, the sparse sensory codes are used to generate behavioral outputs based on learned associations, e.g. to avoid or approach an odor source.

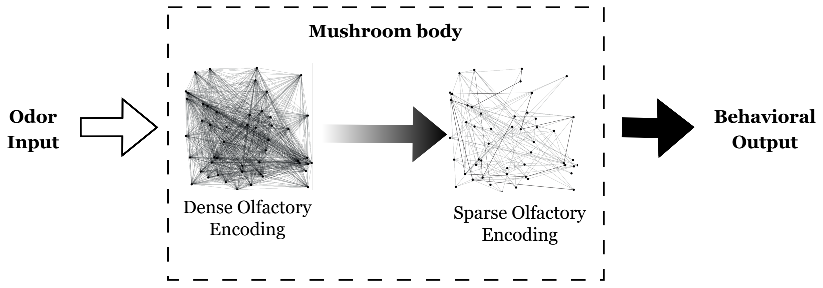
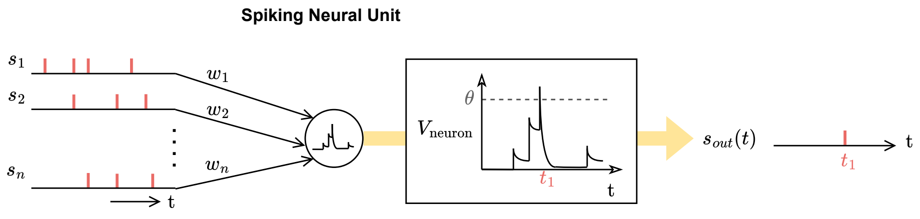


Figure 1.2: Conceptual drawing of the chemo-sensory encoding inside the mushroom body of the *Drosophila*. Each dot represents a neuron that is linked to other neurons. From left to right, the mushroom body receives a dense odor input, which is then transformed into a sparse code. This sparse code takes a smaller area in its neural space, which is beneficial in lowering the brain's power consumption while processing information. In a sparse, olfactory code, the number of connections between neurons is significantly reduced. At the output of the mushroom body, a behavioral output is generated from a sparsely-encoded, sensory cue, and other parameters, which are not included in this drawing.

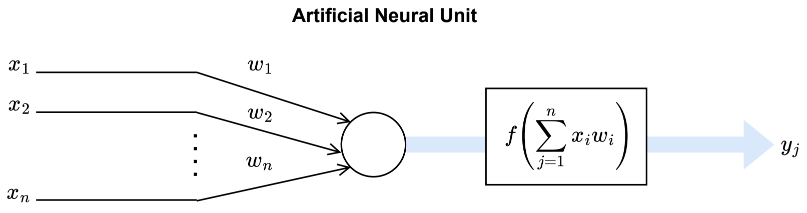
The exploration of these encoding schemes and chemo-sensory capabilities of the *Drosophila* has been extensive. Considerable advancements have been made in identifying fundamental computational principles by modeling olfactory pathways inspired by insects [29, 25, 30]. Various studies have used the model in Figure 1.1 to simulate the olfactory pathway in software [31, 22, 32]. These investigations substantiated and illustrated critical components, such as adaptability and plasticity features in learning. Diamond et al. proposed a bio-inspired classifier within a framework incorporating real-time input sensor data, leveraging sparseness, inhibition, and plasticity as computational primitives. Their design focuses on GPU-based spiking simulations of the *Drosophila's* olfactory pathway [31, 33]. Rapp et al. proposed a bio-inspired model to explore the computational power of a single spiking neuron for numerical cognition tasks with an emphasis on pre-training the neuron using gradient descent [25]. Tuning of this neuron required a large parameter space of approximately 10,0000 parameters. Recently, a more comprehensive, computational model of the *Drosophila* larva has been successfully created in a Python-based simulator for neural networks (Brian 2 [34]), which corroborated homeostatic control in insect-olfactory learning [29, 35]. It is related to the internal, self-regulation of stabilizing the states of cells and neurons in dynamic environments. Although significant advancements have been achieved via software simulations, exploration of hardware implementations for the *Drosophila's* olfactory pathway model remains limited.

Spiking neural networks

Spiking neural networks (SNNs) aim at emulating the way biological neurons exchange sensory information using spikes. Figure 1.3a shows the mechanism of a single spiking neuron, which is different from an artificial neuron used in artificial neural networks (ANNs). In the case of an artificial neuron, as shown in Figure 1.3b, the neuron receives real-valued inputs ($x_1, x_2, ...x_n$) and performs a multiplication with respective weights ($w_1, w_2, ...w_n$). The weighted sum passes through an activation function $f()$. This function determines the final output value of the neuron, denoted as y in Figure 1.3b. The spiking neuron utilizes binary signals to communicate with other neurons. The spiking neuron receives spikes ($s_1, s_2, ...s_n$) and accumulates the spiking information over time t. It then updates its membrane potential (denoted as V_{neuron} in Figure 1.3a) using information from incoming spikes and only outputs a spike s_{out} upon exceeding a threshold value θ, after which it resets its membrane potential. This mechanism models the basic principle of a biological neuron, referred to as the *integrating and firing mechanism*.



(a) A spiking neuron



(b) An artificial neuron

Figure 1.3: Difference between a spiking and an artificial neuron. A spiking neuron receives discrete-valued spike trains ($s_1, s_2, ...s_n$) from preceding neurons at the synapses. The neuron then performs calculations using the spiking input to update its neural membrane potential (V_{neuron}). If the membrane potential exceeds a certain threshold value θ, the neuron will fire an action potential, encoded as a spike at its output ($s_{out}(t)$) which is transmitted to other neurons. In this figure, the spiking neuron outputs a spike at time t_1 upon exceeding the threshold. The artificial neuron, on the other hand, is a simplified neural model that attempts to conceptually mimic a biological neuron in its simplest form. The neuron multiplies the received inputs ($x_1, x_2, ...x_n$) with their respective weights ($w_1, w_2, ...w_n$) and consequently sums them. This result is passed on to an activation function (f) which determines the output value of the neuron (y).

Neuromorphic implementations of the *Drosophila* olfactory pathway

The first silicon-based, neuromorphic olfaction chip successfully used an SNN [36]. This made an advancement since the first design of an analog electronic nose without an SNN, introduced by Persaud and Dodd in 1982 [37]. The last decade has witnessed a rise in attempts to simulate the olfactory pathway of the *Drosophila*, but has been limited to odor detection and classification tasks using (supervised) offline training. For example, a *Drosophila*-inspired, supervised odor classification model was built on the mixed-signal, *Spikey* chip and proposed a proof-of-concept for its usage in generic multivariate classification, e.g. using the Modified National Institute of Standards and Technology (MNIST) database [33]. However, this model is limited to offline training.

Little work has been dedicated to the exploration of the computational and processing capabilities of the low-footprint *Drosophila* olfactory pathway applied to learning applications at the edge. Systems in edge processing applications require online learning capabilities on highly variable and contextual data. The computing platforms in the majority of olfactory-inspired studies are software simulators [25, 30, 29, 35]. Even though software provides the convenient flexibility to build and simulate models, hardware - and specifically silicon-based solutions - can closely emulate the actual electrical behavior of a biological model [4]. However, analog and mixed-circuit designs require a complex and relatively long design time, simultaneously subjected to process, voltage, and temperature (PVT) variations, devices' mismatches, and low programmability [38, 10]. In the realm of neuromorphic computing and insect-inspired olfactory emulation, several digital hardware designs have been proposed, for example, using Intel's Loihi neuromorphic chip [39, 40], and FPGA-based platforms [41, 10]. Digital circuit designs are advantageous to accelerate benchmarking and prototyping of neuromorphic computing hardware [10].

Figure 1.4 presents an overview of existing hardware and software designs related to olfactory pathway modeling ranging from a top-down approach for exploring computational primitives to a bottom-up perspective of mimicking the human nose [10]. The figure shows the relative position of the bio-plausibility level of the proposed solutions against the number of neurons in the olfactory network.

Within the scope of this thesis, the main target is to evaluate bio-plausibility for small neural networks. The bio-plausibility level of a proposed model can be ranked using the following four criteria:

1. The model uses a spiking neural network, and utilizes spikes for the inter-neural communication [4, 45].
2. The model implements a three-layered network identical to the anatomical pathway of a *Drosophila* [46, 47, 19].
3. The model attempts to reproduce and implement computing rules for learning that experimental evidence infers [48, 49].
4. The chosen neuron model exhibits features (synaptic plasticity, inhibition, spike frequency adaptation and neuromodulation [1]) close to the biological neuron [50].

The model of a larval *Drosophila* has the highest attainable score in biological plausibility [15, 16] for a small network. Although, there are many other smaller organisms with even a lower neural count, we focus on the *Drosophila* larva. For the sake of completeness, Figure 1.4 also contains designs inspired by the mammalian, olfactory bulb. This is also a well-studied model, but its neuron count is at least 100 times larger than for the *Drosophila*, making them unsuitable for small-scale networks.

[1]These concepts will be further elaborated in Chapter 2.

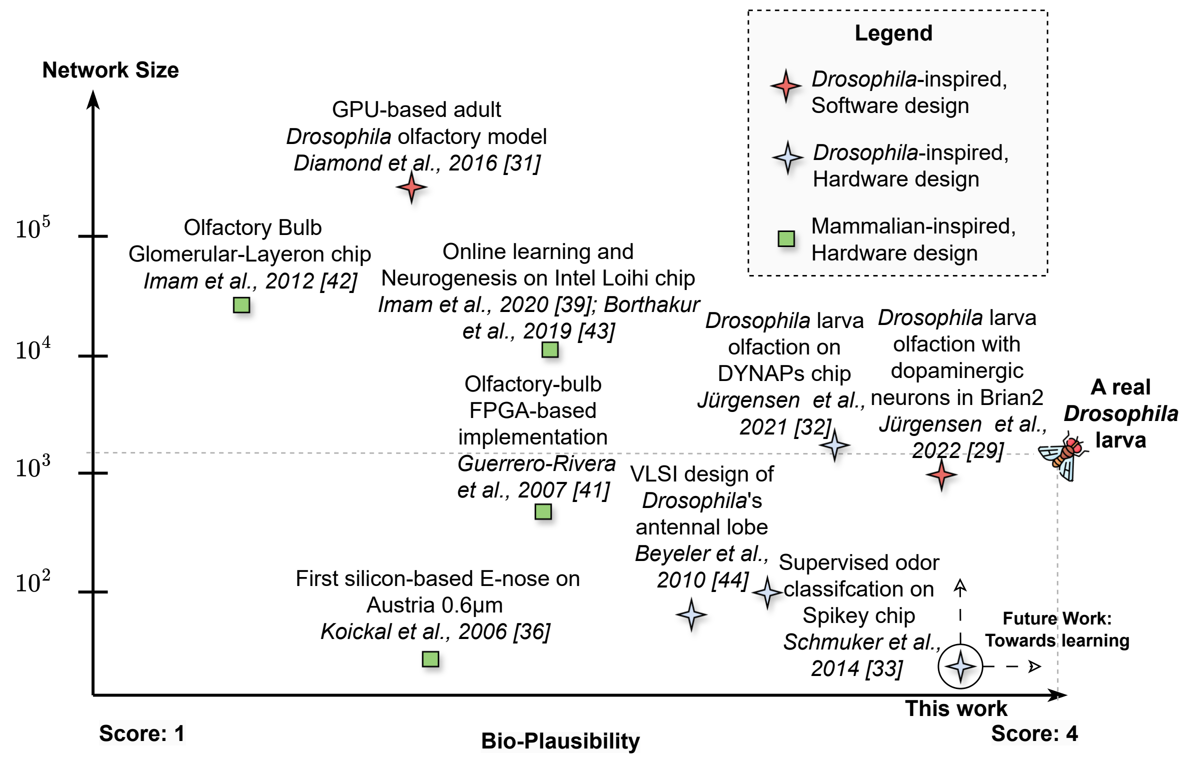


Figure 1.4: Overview of proposed designs of olfactory pathway models from mammals and insects. The solutions are classified in their level of relative bio-plausibility and the number of neurons. The designs from the figure proposed solutions to mimic the computational principles and the neural mechanisms of the olfactory pathway in an odor detection application with offline training [42, 31, 39, 43, 41, 36, 44, 33, 29, 32]. This overview is non-exhaustive and only includes models using SNNs.

Research gap

Although several studies have investigated how olfactory pathway models can be remade with SNNs on hardware platforms, only few designs have focused on creating a bio-plausible hardware implementation using the key features of prototyping a digital design, such as modularity, adaptability and programmability [41]. Therefore, this thesis proposes a design for a neuromorphic, FPGA-based hardware emulator of the larval *Drosophila* olfactory pathway model as seen in Figure 1.1. We aim at scoring high in the bio-plausibility level by satisfying our criteria list. We also propose a modular, neuromorphic framework taking inspiration from existing, successful emulations of this pathway model, used in odor detection. Despite the large number of studies investigating bio-plausible emulations of chemical detection for artificial and natural odors, most models overlook the efficacy of the olfactory neural design in dynamic environments. This thesis, however, addresses this issue by introducing a strategy for bio-plausible emulation of the olfactory pathway model of the *Drosophila* larva. To put it briefly, the research scope of the thesis can be formulated in the following main research question:

Main Research Question

How can the olfactory pathway model of a *Drosophila Melanogaster* larva support context-dependent processing *at the edge* using neuromorphic hardware?

The main research question is divided into four sub-questions:

1. *Which biologically inspired neuron model could be used to closely emulate the spatio-temporal dependencies of neurons in the Drosophila olfactory pathway model*?
2. *What type of digital circuitry is able to simulate the neurons in the Drosophila olfactory pathway model while maintaining accuracy?*
3. *How do time-multiplexed, hardware attributes impact the granularity of the olfactory pathway model*?
4. *How can bio-plausibility be maintained at a footprint that is acceptable for edge computing applications?*

The scope of this thesis addresses these sub-questions, at the exception of the latter. The fourth sub-question requires a complementary study, which is left for future work. Ultimately, the contribution of the thesis is three-fold:

1. We propose a bio-plausible SNN inspired by the larval *Drosophila* olfactory pathway, which relies on time-stepped simulation using the Lava framework toward solving context-dependent tasks.
2. We present an FPGA-based hardware design of a bio-plausible neuron, relying on a digital solver.
3. We introduce a strategy to utilize our modular design for prototyping olfactory-inspired neurons on hardware.

Thesis outline
The rest of the thesis is organized as follows. Chapter 2 provides the necessary background to build a fundamental understanding of topics related to the thesis and the research. Firstly, this chapter explains the relevant terminology in neuroscience, including the anatomy of the olfactory pathway model and neuronal principles. Chapter 3 presents the proposed solution to the research question in two stages: (i) software design and (ii) hardware design. Then, in Chapter 4, the results of the design is presented and evaluated. Lastly, in Chapter 5, we present the conclusions of this thesis, summarizing its contributions and elaborating on future work.

2
Background

This chapter provides the context underlying the thesis research to establish an understanding of the proposed solution. Section 2.1 describes the principles of biological neurons, introducing the anatomy and fundamental concepts. Section 2.3 introduces the olfactory pathway model of the *Drosophila*, which is divided into (I) anatomy, (II) sensory pre-processing, and (II) olfactory learning. Then, Section 2.2 elaborates upon spiking neuron models that are relevant to the modeling of the olfactory pathway model. Finally, Section 2.4 outlines proposed designs inspired by the olfactory pathway model of the *Drosophila* larva to understand the state-of-the-art.

2.1. Principles of biological neurons

This section introduces the principles of biological neurons, starting with the anatomy of the neuron. Then, the working principle of the neuron is explained. Lastly, cellular mechanisms such as synaptic plasticity and spike frequency adaptation are introduced to build a comprehensive understanding of the neuron on both a cellular and mechanistic level.

2.1.1. Anatomy of the neuron

The biological neuron is the fundamental building block of any sensory system. Neurons use biochemical reactions, allowing for signal processing and transmission. There are three main neuron types: (1) sensory/input neurons, (2) motor/output neurons, and (3) interneurons. Firstly, the sensory neurons are activated by sensory stimuli, e.g., odors. For example, the input neurons involved in olfaction are called olfactory receptor neurons and are located in the antennae of the fruit fly. Secondly, output neurons are responsible for transmitting signals out of the brain, for example, to exert movement. Thirdly, interneurons establish communication between the input neurons and output neurons. Typically, input neurons are not near output neurons, and therefore these interneurons facilitate relays of signals to reach sensory or motor parts [51, 52].

Neurons transmit sensory information throughout the brain and nervous systems of organisms. A biological neuron consists of three main parts: dendrites, the soma and the axon [53, 54]. Figure 2.1 shows what a typical neuron looks like and highlights these compounds of the neuron.

- **The soma** computes the state of the neuron expressed as the membrane potential. A neuron only has one soma, the central processing unit, processing the receiving signals. The soma generates a pulse as an output signal, i.e. an action potential or ”spike” if its membrane potential exceeds a certain threshold (commonly denoted as θ). The action potential is an impulse where the membrane potential rapidly rises and falls, and causes a change in electric polarity across the (membrane of) the axon.

- **Dendrites** collect the signals from other neurons and send them to the soma of the neuron. Dendrites have (dendritic) branches with tiny protrusions, which we refer to as ”claws”. This is visible in Figure 2.1.
- **The axon** is in charge of transmitting this output signal from the soma to other neurons.

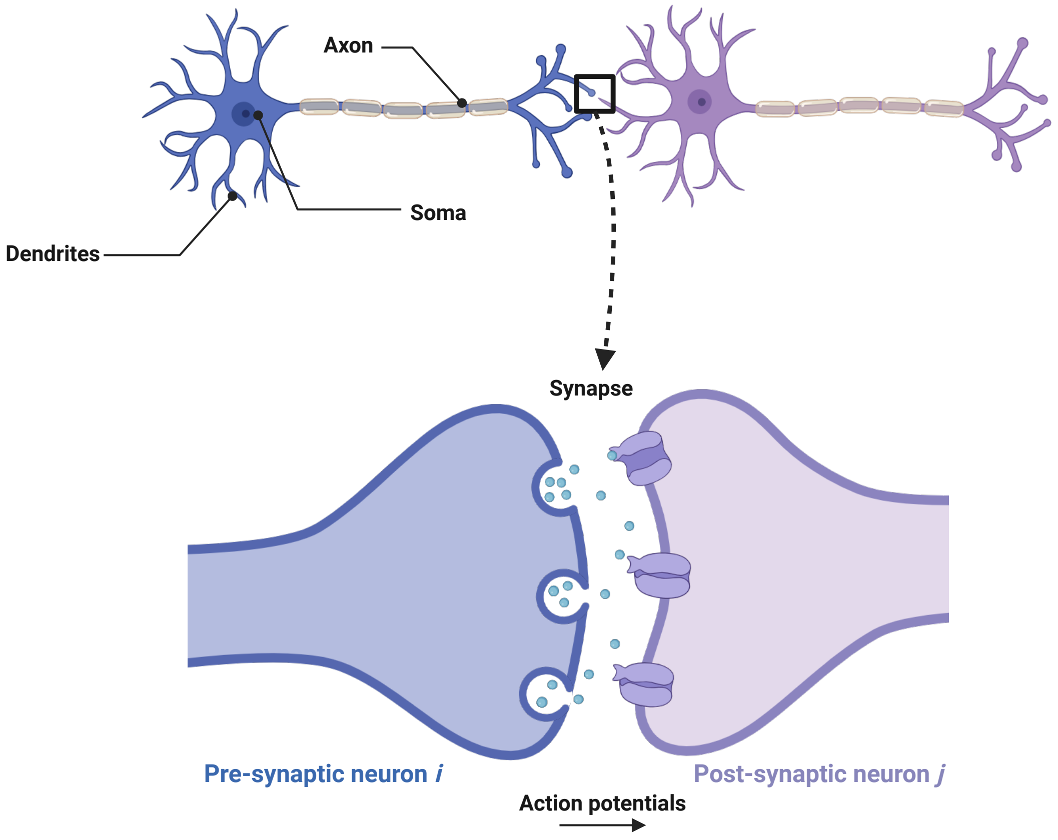


Figure 2.1: Conceptual drawing of a pre-synaptic neuron i and a post-synaptic neuron j. The dendrites, the soma, and the axon of the pre-synaptic neuron *i* are indicated in the figure. The action potential from pre-synaptic neuron *i* propagates to its axon terminal, where neurotransmitters are released into the synapse. The neurotransmitters bind onto the cell membrane of the dendrites of the post-synaptic neuron *j*. In other words, this is how the pre-synaptic neuron transmits the action potential to the post-synaptic neuron [53]. The illustration was made with BioRender.

Communication between neurons is established through synapses (indicated in Figure 2.1), which are small gaps between the axon of the neuron and a dendritic branch of a succeeding neuron. Typically, a single neuron in a vertebrate cortex arborizes with more than 10^4 post-synaptic neurons [54]. The human brain contains approximately 100 billion neurons with 100 trillion synapses [55].

2.1.2. Membrane potential dynamics

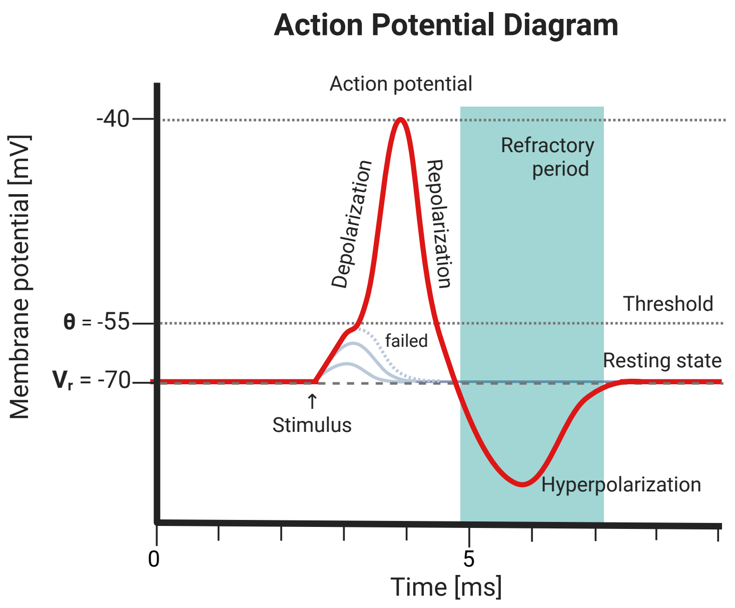


Figure 2.2: Action potential in a post-synaptic neuron A pre-synaptic stimulus releases neurotransmitters to the synapse, which causes a depolarization in the post-synaptic neuron. The depolarization accumulates for every incoming spike until the membrane potential of the post-synaptic neuron crosses its threshold voltage θ. Then, the post-synaptic neuron elicits a spike, where its membrane potential rapidly rises and falls with a total duration of about 1 to 2 ms [53]. The neuron quickly depolarizes due to a large ionic inflow from the pre-synaptic neurotransmitters into the channels of the post-synaptic neuron. Then, the neuron quickly repolarizes, returning to its resting potential V_r. Due to a delay in closing the (voltage-dependent potassium) channels of the post-synaptic neuron, the neuron hyperpolarizes. This is internally resolved whereby the post-synaptic neuron eventually returns to its resting state. The figure also shows a refractory period during which the neuron cannot evoke another action potential.

The membrane potential dynamics is generalized as the generation of action potentials due to the release of neurotransmitters in the synapses. Figure 2.2 shows how a post-synaptic neuron creates an action potential. The neuron normally rests, whereby the membrane potential stays at the resting potential V_r. When a pre-synaptic neuron emits a spike train, neurotransmitters are released in the synapse. The post-synaptic neuron receives neurotransmitters through channels residing at the cell membrane (indicated in Figure 2.1), inducing an ionic influx (i.e. an electric current). Consequently, the change in incoming, electric currents influences the membrane potential of the post-synaptic neuron [53]. This causes the neuron to rapidly depolarize and repolarize within 1 or 2 ms [53]. This spike traverses the axon, which - in turn - releases neurotransmitters in the synapses with other downstream neurons. Figure 2.2 also indicates a so-called refractory period during which the neuron cannot generate another action potential. This typically happens when the neuron requires more time to re-stabilize after spiking, which causes an undershoot, i.e. hyperpolarization. As the magnitude and the time course are approximately the same for every action potential, the information is carried in the number and timing of the spikes rather than the form of the action potential itself [54].

2.1.3. Cellular mechanisms

Neurotransmitters play a crucial role in various physiological processes - including cognition and emotion - and autonomic functions such as heart rate. Neurotransmitters facilitate communication between neurons by relaying and modulating the signals between pre- and post-synaptic neurons upon their secretion (exocytosis). In other words, they may inhibit or encourage the firing of the post-synaptic neuron. There are different neurotransmitters, which are generally classified into two classes: excitatory (e.g. glutamate [56]) or inhibitory (e.g. γ-aminobutyric acid [57]). However, some neurotransmitters may be either, e.g. dopamine [58]. Two important cellular mechanisms moderate the release of neurotransmitters, influencing the excitability of neurons (1) on synaptic level through *synaptic plasticity* and (2) on neuron-level with *spike-frequency adaptation (SFA)*.

Synaptic plasticity

Within a synapse, the neurotransmitters can undergo modifications to strengthen or weaken the signal transmission between the pre- and post-synaptic neurons. This plays a prominent role in the memory and learning of neural networks, commonly referred to as *synaptic plasticity* [59]. The modulation of the synaptic transmission may vary in duration ranging from milliseconds to days, which gives rise to the following categories:

1. **Short-term synaptic plasticity**: Synaptic plasticity lasting tens of milliseconds to minutes is classified as *short-term plasticity* and may induce inhibition (short-term depression) or excitation (short-term potentiation) of post-synaptic neurons [60, 59]. Short-term depression and synaptic facilitation in a post-synaptic neuron are both mainly influenced by pre-synaptic calcium (Ca^+) influx [61, 62].
2. **Long-term synaptic plasticity**: Long-term synaptic plasticity i.e. synaptic modification which lasts for hours and even days, is formed by an ensemble of strongly coupled neurons that encode a specific event. This is typically associated with the formation of memory. In this case, we discern the inhibitory and excitatory effects, i.e. depression (LTD) or potentiation (LTP), respectively [63]. Traditionally, there are various examples of neurotransmitters, which facilitate long-term memory through *neuromodulation*, e.g. acetylcholine and noradrenaline. However, dopamine has been widely associated with two-fold, long-term synaptic plasticity [64]. Dopamine mediation plays a crucial role in the learning ability of organisms to predict future events based on experiences and continual environmental inputs [65].

Spike frequency adaptation

In contrast to the synaptic plasticity mechanisms mentioned above, SFA addresses an individual neuron rather than a collection of neurons. It plays a crucial role in neural information processing which is largely influenced by three main ionic currents, i.e. the adaptation currents [66]. These are M-type currents [67], afterhyperpolarization (AHP)-type currents [68], and sodium (Na^+) currents [69]. These adaptation currents accommodate the neuron's excitability and act in an inhibitory manner to limit further action potential discharge upon neuron activation, decreasing the firing activity of the respective neuron [68]. Due to SFA, the firing frequency of the neurons decays to some steady-state value (f_∞), which is approximated by an effective adaptation time constant (τ_{eff}), in the range from milliseconds to seconds [66]. Short-term depression at the synapse vesicle pool (the bulbs in the axon terminal in Figure 2.1) drives the neurotransmitters in a depression state. During this phase, the signals from the pre-synaptic neuron cannot be transmitted through the synapse to the post-synaptic neuron [70, 71]. SFA improves the efficiency of sensory information [72] and consequently optimizes its transmission [73]. Figure 2.3 illustrates the SFA mechanism, whereby a sustained input stimuli is given to a neuron with and without SFA.

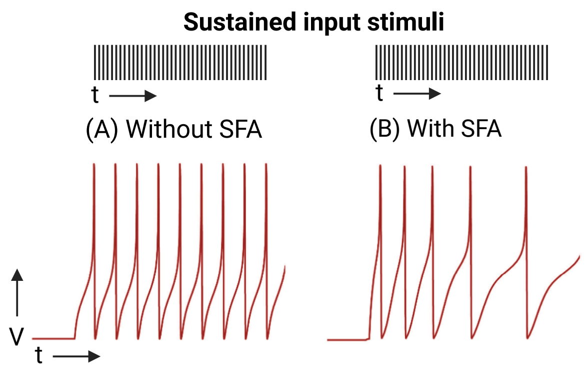


Figure 2.3: SFA mechanism We assume that in both cases, the same inputs spikes are given. (A) A neuron without SFA exhibits regular spiking to a sustained spiking stimulus. (B) A neuron with SFA that has the same input, will decay its spiking rate for a sustained stimulus. Its decay in spiking rate will decay to a steady-state value, τ_{eff}.

2.2. Spiking neuron models

A spiking neuron model is a mathematical characterization of biological neurons, by modeling their attributes and mechanisms. These models typically contain at least the soma, which is considered to be the computing centre [50]. The outline of spiking neuron models, modeling the dynamics of the neuron and its spatial structures (i.e., synapses and dendrites), range from sophisticated conductance-based models (e.g. Hodgkin-Huxley) to generalized point-neuron models (e.g. leaky-integrate-and-fire neuron) mimicking the minimum attributes of the neuron. In synaptic modeling, we distinguish between current-based and conductance-based synapses. The former aims at modelling ionic current with a fixed amplitude. The latter computationally models the mechanisms from the ionic changes in the dendrites [50]. Conductance-based neuron models are more advanced and biophysically accurate models but are computationally expensive exhaustive due to solving coupled partial differential equations. We will compare the more advanced conductance-based design, modelling both the synapses and the dendrites, with a LIF neuron with conductance-based synapses, excluding the dendritic structures.

2.2.1. Synaptic modelling

The synapses capture the dynamics of the neurotransmitters propagating from the pre-synaptic neuron to post-synaptic neurons. The released neurotransmitters from post-synaptic neurons generate an ionic charge due to the potential difference between the interior and exterior of the cell. This is caused by the movements of ions in the synapse, which is modeled as a current $I(t)$:

$$I(t) = C_m \frac{V_r - V(t)}{dt} \tag{2.1}$$

where C_m denotes the cell membrane capacitance, $V(t)$ is the membrane potential at a time t, and V_r is the resting potential of the neuron. In order to solve the membrane potential, an integration over time must be computed. This is covered in Section 3.1, focusing on the actual compuations of the neuron.

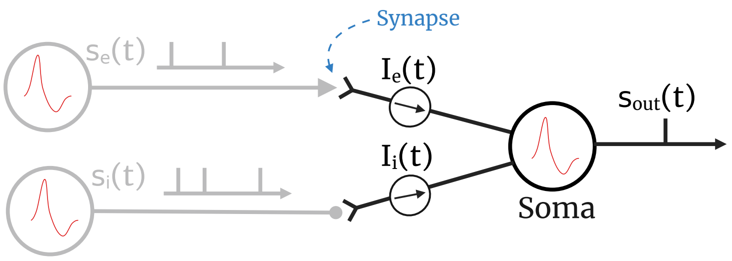


Figure 2.4: Spiking neuron with two synapses The neuron has two dendrites, linked to a pre-synaptic excitatory and inhibitory neuron. Note that in this case, the contribution of the dendrite is neglected, analog to simply being an ideal wire with zero resistance. The spike trains from both pre-synaptic neurons ($s_e(t)$ and $s_i(t)$) are sent to the soma. The soma computes the membrane potential using the incoming spike trains and currents ($I_e(t)$ and $I_i(t)$). It generates a spike train $s_{out}(t)$ upon exceeding the threshold.

Figure 2.4 shows a neuron with one excitatory and one inhibitory input current (denoted as $I_{e,1}(t)$ and $I_{i,1}(t)$, respectively). The synaptic inputs from pre-synaptic neurons mediate in the excitation and inhibition of post-synaptic neurons, thus providing an excitatory current ($I_e(t)$) or an inhibitory current ($I_i(t)$), respectively. These currents influence the membrane potential of the post-synaptic neuron.

- **Current-based (CUBA) synapses**: Current-based synapses model the ionic charge as an ionic current with a fixed amplitude. In this case, the product of the potential difference and the capacitance in Equation 2.1 is assumed to be a parameter with a fixed value, which is referred to as a fixed synaptic current efficacy J_{syn} expressed in Ampere [74], yielding

  $$I^{CUBA}(t) = J_{syn} s_{syn}(t), \tag{2.2}$$

  where $s_{syn}(t)$ is a function that models the time course of pre-synaptic spiking occurrence [74, 50], similar to an exponential time constant.

- **Conductance-based (COBA) synapses**: Conductance-based synapses consider that ionic currents depend on the conductance of the different channels. Furthermore, in conductance-based synaptic modeling, the ionic current follows from multiplying with a time-varying potential difference between the interior and exterior of the cell (soma), rather than a fixed value. Conductance-based synapses describe the synaptic dynamics in a more biologically accurate fashion [50]

  $$I^{COBA}(t) = g_{syn} s_{syn}(t)(V_{syn} - V(t)), \tag{2.3}$$

where g_{syn} and V_{syn} are respectively the conductance and the reversal potential of the synapse.

The difference between $I^{CUBA}(t)$ in Equation 2.2 and $I^{COBA}(t)$ in Equation 2.3 lies in whether they account for or omit the temporal influence of ionic currents.

2.2.2. Conductance-based spiking neuron models

Conductance-based models are biophysically plausible models whereby the level of resemblance with biological neurons may be adjusted according to the research scope from close biophysical modeling with the Hodgkin-Huxley neuron to simplified leaky integrate-and-fire neurons.

Hodgkin-Huxley

The Hodgkin-Huxley (HH) model is a biophysical interpretation of neurons that closely examines the ionic dynamics, by modeling a neuron's soma, dendrites, and synapses [75]. In this case, the ionic current is related to the currents created by sodium (I_{Na}) and potassium (I_K) ionic charge. Furthermore, a leakage term is included that models the passive properties of the neuron cell, I_L. Therefore, the total ionic current $I(t)$ in a Hodgkin-Huxley is modeled as:

$$I(t) = I_{Na}(t) + I_K(t) + I_L(t) \tag{2.4}$$

The three terms from Equation 2.4 are further dissected and equated with the definition of the total current of the neuron in Equation 2.1 revealing the conductances (g_{Na}, g_K, g_L), and the time-varying potential difference for each ion channel:

$$I(t) = C_m \frac{dV(t)}{dt} = g_{Na} m^3 h (V(t) - V_{Na}) + g_K n^4 (V(t) - V_K) + g_L (V(t) - V_L) \tag{2.5}$$

where $V(t)$ is the membrane potential of the post-synaptic neuron at time t. Dendritic modeling is covered by the variables m and n, which are related to sodium and potassium activation, and h is the sodium inactivation variable. These variables vary between 0 and 1 and are dimensionless. They represent the time course of the action potential, which is governed by differential and subsidiary equations [75]. The resting potential of the sodium and potassium channels are V_{Na} and V_K, respectively. The leakage potential of the neuron is denoted as V_L. Equation 2.5 involves integration over time to compute the membrane potential at any given time. Consequently, this constitutes a system of four-dimensional ordinary differential equations [76, 50]. The Hodgkin-Huxley model is an elegant model that accurately models neurons in their sub-threshold regime, which requires extensive computational resources. However, it is possible to approximate the (sub-threshold) dynamics of the HH model. Specifically, in research where a detailed (ionic-based) neuron model is unnecessary, the HH model can be approximated by a simplified conductance-based neuron model using the dynamics of conductance-based synapses. This leads to a plausible combination of one of the simplest, point-neuron models - called the leaky-integrate-and-fire neuron - with conductance-based synapses [77, 50].

LIF neuron with conductance-based synapses

The LIF neuron was introduced by Louis Lapicque in 1907 [78, 79]. The biological neuron is modeled as a point model (without dendrites) with a membrane that is described as an RC circuit, where R is the cell resistance and C is the membrane capacitance. The structure is *leaky* to represent the ion diffusion through the membrane, which has previously been used in equations as the leaky current I_L. Figure 2.5 shows an equivalent circuit of a LIF neuron with conductance-based synapses (COBALIF neuron). In the figure, the inverse of the resistance, *i.e.* conductance g, is used in labeling resistors.

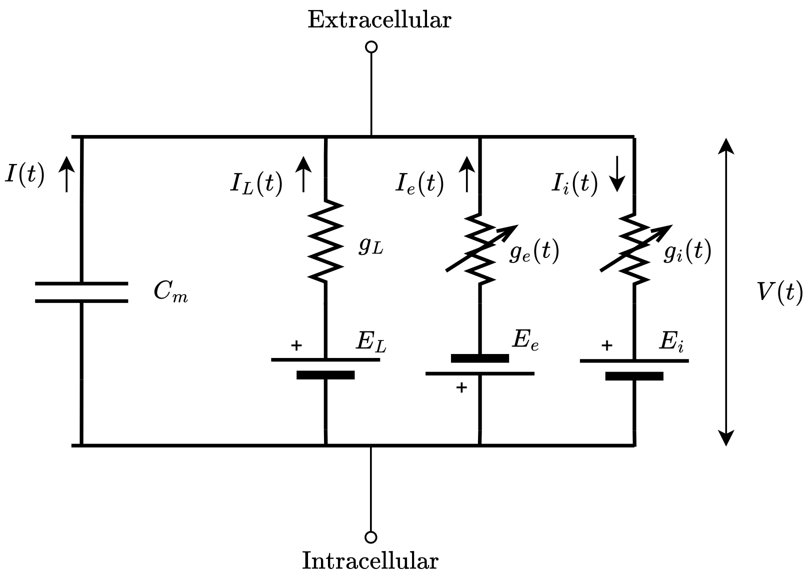


Figure 2.5: Equivalent circuit of a LIF neuron with conductance-based synapses (COBALIF neuron) point-conductance model containing synaptic conductances. This includes the time-varying synaptic conductances. C_m is the membrane capacitance. $I(t)$ is the total current.

The circuit in Figure 2.5 is similar to the electrical circuit that describes the Hodgkin-Huxley model [75]. However, in this case, the ionic, injected sodium and potassium current (I_{Na} and I_K) are approximated by excitatory and inhibitory synaptic currents (I_e and I_i) [75, 50]. This model approximates each component of the total ionic current as the conductance of the ion with the difference between the membrane potential and the respective ion. For this neuron model, the total current $I(t)$ in Equation 2.5 simplifies to a non-ionic specific description:

$$C_m \frac{dV}{dt} = g_e(E_e - V(t)) - g_i(E_i - V(t)) + g_L(E_L - V(t)) \tag{2.6}$$

where the g_e and g_i are the excitatory and inhibitory conductances, respectively. The resting potentials for each term in the equation are denoted as E_e, E_i and E_L, whereby it holds that $E_i \leq V_r < E_e$ [50]. The leakage potential falls within the same regime but is typically not constrained in the same fashion. These potentials arise from the equilibrium potentials of the ion channels, which introduce non-linearity to the synaptic input summation. The reversal potentials of the excitatory (E_e) and inhibitory conductance E_i) are 0 mV and -75 mV, respectively [80]. The parameters g_e and g_i represent the non-negative excitatory and inhibitory conductances, which are small in magnitude and non-negative in this model [50]. Figure 2.6 illustrates how a neuron generates spike upon integrating the incoming spikes from a pre-synaptic neuron, which is governed by Equation 2.6. A pre-synaptic neuron sends spikes that cause short excitatory current pulses (denoted as I_e). For each spike, the membrane potential augments and approaches the threshold potential (V_{th}). The neuron triggers a spike when the membrane potential reaches a firing threshold V_{th}. The neuron then immediately discharges its current to arrive at its resting potential V_r. The neuron then generates an output spike. Afterward, the neuron enters its refractory state during which the potential of the neuron is at rest and cannot spike, which lasts for a refractory period τ_r. The LIF neuron with conductance-based synapses as described above has been used in various spiking neural networks, modeling the olfactory pathway model of *Drosophila* larvae [25, 29, 32].

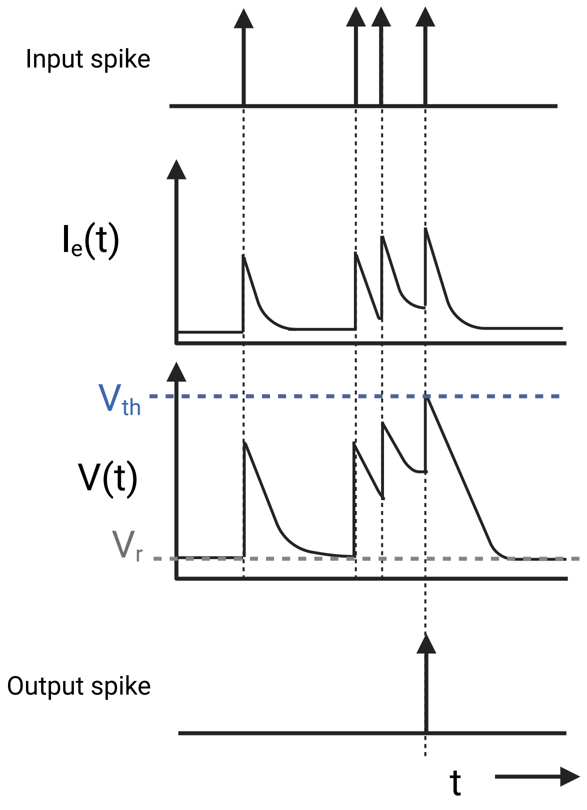


Figure 2.6: Generation of a spike. Each spike that arrives at the neuron causes a short current pulse of which the voltage decays exponentially. In this case, only the excitatory current is included, which follows from the multiplication between of the excitatory conductance and the potential difference. The other elements from Equation 2.6 are excluded. The membrane potential follows from integrating Equation 2.6. Only one spike is generated by the neuron, which happens when the membrane potential $V(t)$ exceeds the threshold voltage V_{th}. The neuron then is in its refractory state for a given refractory period τ_r and the membrane potential is at its resting potential V_t.

2.3. The *Drosophila* olfactory pathway model

This section is focused on giving background of (i) the connectome of the *Drosophila* larva, (ii) olfactory pre-processing and (iii) olfactory learning and memory. Firstly, we will cover the identity and functionalities of the neurons in more detail. Secondly, the olfactory processing is discussed which explains how olfactory sensory information is processed according to a four-stage procesing. Furthermore, this subsection is dedicated to explaining olfactory learning and memory. We will examine the olfactory pathway model by firstly introducing the underlying sensory (pre-)processing pipeline. Finally, olfactory learning and memory will be explained, which happens after the pre-processing. This involves the understanding of the mechanisms that are related to learning, memorizing, and recalling odors. Furthermore, we will explain how outputs of this pathway model are shaped.

2.3.1. *Drosophila* larva connectome

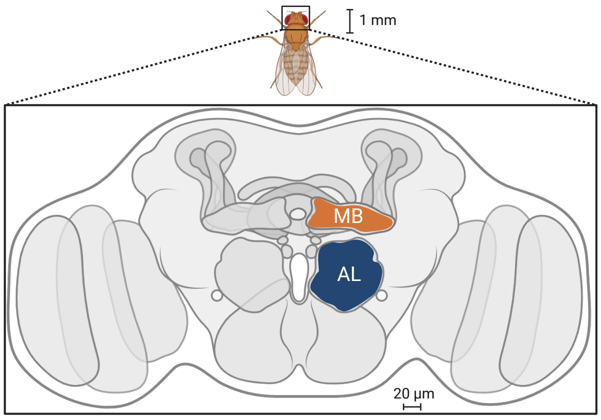


Figure 2.7: Illustration of the anatomy of an adult fly brain This figure illustrates a female adult fly brain. The antennal lobe (AL) - in blue - and the mushroom body (MB) - in orange - are highlighted and labeled, for one hemisphere. The illustration is inspired by Aso et al [81].

In the past decade, the study of the olfactory nervous system has significantly progressed to the point that many characteristics of its anatomical structure and mechanisms are identified [82, 83]. The processing pipeline of olfactory information in a mammal corresponds with the one from the *Drosophila* larva [84]. The olfactory pathway of the *Drosophila* employs a layered architecture to process odors, which is similar to the olfactory bulb of vertebrates [85, 86, 46]. However, the connectome of the mammal consists of millions of neurons and constitutes a large network [13]. The connectome - a comprehensive collection of neural pathways in the brain - of the insect brain is approximately 331,565 $\times$smaller. With about 3,016 neurons and 548,000 synapses, the connectome of a *Drosophila* larva is one of the most well-studied small brains to date [14]. Figure 2.7 shows an illustration of the fly brain is shown, where two important regions are highlighted: the antennal lobe (A) and the mushroom body (MB). These regions are involved in processing sensory information, facilitated by a network of neurons.The olfactory pathway model consists of different neuron types, which are characterized by their types of neurotransmitters (i.e. excitatory or inhibitory) and their functionalities in processing of olfactory information. Figure 2.8 shows the connectome of the *Drosophila* larva for olfactory processing. This figure shows the neurons, their location in the brain, and connections within the network, and the various processing stages. An overview of the different neuron tupes are presented in Table 2.1, covering the total count within a region, the neurotransmitter type and the respective processing stage.

Table 2.1: Overview of the neuron types in the olfactory pathway model.

Processing stage	Neuron type	Abbreviation	Region	Total count	Neurotransmitter type	
					Excitatory	*Inhibitory*
I	**Olfactory receptor neuron**	ORN	Antenna	21	✓	
II	**Projection neuron**	PN	AL	21	✓	
	Local interneuron	LN	AL	21		✓
III	**Anterior paired lateral neuron**	APL neuron	MB	1		✓
	Kenyon cell	KC	MB	72	✓	
IV	**Dopaminergic neuron**	DAN	MB	2	✓	✓
	Mushroom body interneuron	MBIN	MB	2	✓	✓
	Mushroom body output neuron	MBON	MB	2	✓	✓

The olfactory pathway model starts at the sensory periphery, where the antennas are located. The olfactory receptor neurons (ORNs) in the antennas detect odors and transmit the sensory information encoded as action potentials to secondary processing centers. The secondary processing centers consist of the (i) antennal lobe (AL) and (ii) the mushroom body (MB). Inside the antennal lobe, there are two types of interneurons: the local interneurons (LNs) and the projection neurons (PNs). Their synaptic interactions with the ORNs form the first stage in processing odor information [15]. Lastly, the mushroom body is the main olfactory processing center where odors are learned and memorized. In the mushroom body, stimuli-specific activation of Kenyon cells (KCs) plays a prominent role in efficient odor-encoding [48]. Then, the output neurons of the mushroom body (MBONs) transmit the output pulses which induce behavioral actions. Generally, the MBONs do not directly encode the odor, but rather a negative or positive valence, anticipating the behavioural response, i.e avoidance and approach, respectively [49, 84, 81].

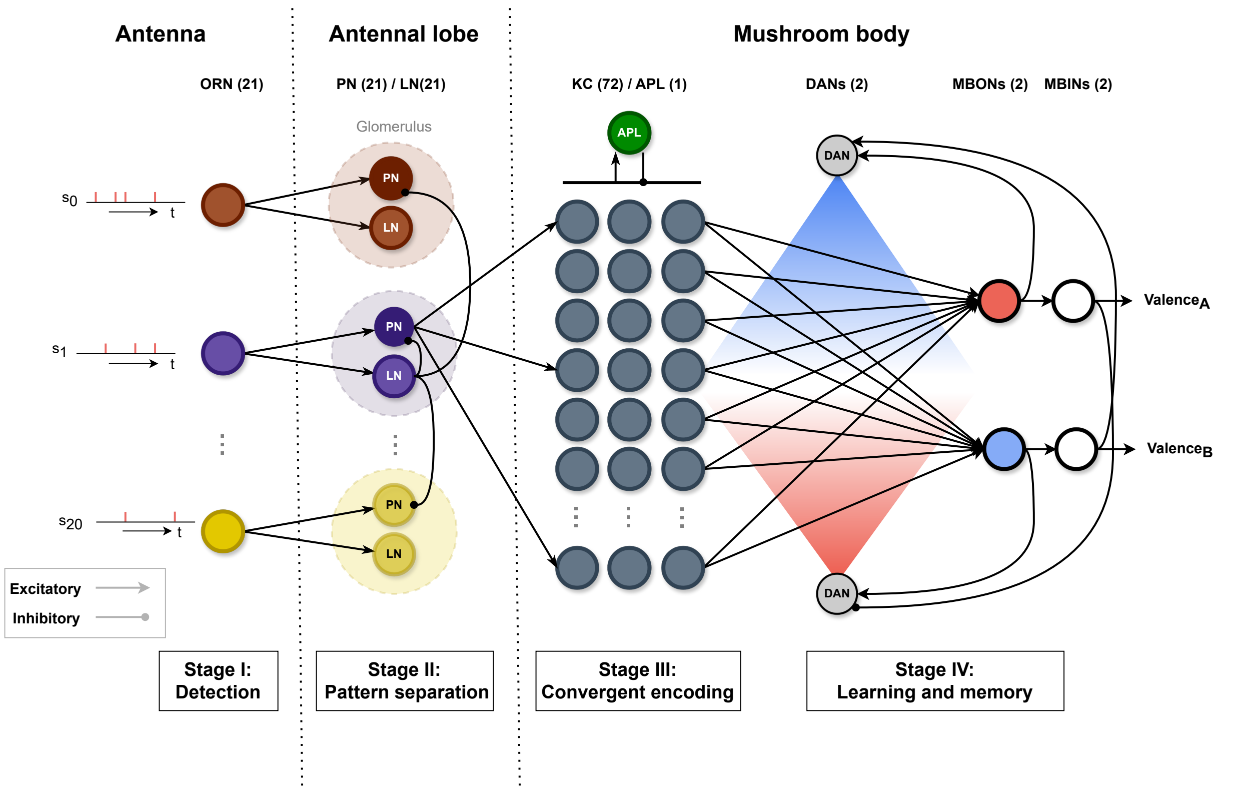


Figure 2.8: *Drosophila* larva connectome and the olfactory pathway model From left to right, the figure shows the olfactory pathway model starting from the antenna to the mushroom body. In the antenna, 21 ORNs each receive a unique train of stimuli from sensing the odors. The ORNs then project to their cognate LNs and PNs within a glomerulus. In the antennal lobe, 21 pairs of LNs and PNs innervate with the ORNs where lateral inhibition is employed by the LNs. This builds pattern separation. Then, the projection neurons randomly project onto a subset of KCs. The feedback inhibition between the APL and the KCs through all-to-one excitation and one-to-all inhibition. The figure shows the readout and modulatory component of the mushroom body, indicated by the MBONs and the DANs, respectively. The MBONs send out signals which either evoke a Valence A (negative) or Valence B (positive) signal. The connections are returned to the modulatory DANs, which facilitate reward-punishment modulation.

2.3.2. Olfactory sensory pre-processing

Sensory information is typically distributed in nervous systems with large neural networks and advances through multiple processing layers. Consequently, stimulus-specific sensory information is transformed into a useful format that can be used to generate behavioral outputs [87, 88]. Downstream sensory coding governs the mapping and transformation of highly dense stimulus-specific information to a limited number of behavioral outputs [88]. The *Drosophila* learns a given odor through encoding the sensory information, which is transformed throughout the stages and finally represented by only a few activated Kenyon cells in the mushroom body. The activation and association between an odor and the Kenyon cells are unique to each fly. The fruit fly learns through its own experiences what a given odor means and learns to associate a given odor with a behavioral output on a reward-punishment basis [84]. We will cover the stages from input to readout to understand the neural circuitry supporting this sophisticated form of odor encoding. The connectome of *Drosophila* larva for one hemisphere is shown in Figure 2.8. The stages related to olfactory pre-processing are labeled as follows: (I) Detection, (II) Pattern Separation, (III) Convergent Encoding.

Stage I - Detection
A smell is a mixture of many chemical compounds, called odorants. When an insect detects an odor, the odorants bind to the cell surface proteins of olfactory receptors, which are located on the fly's antennas. These olfactory receptors are the chemoreceptors, expressed in the cell membranes of olfactory receptor neurons (ORNs), and are responsible for the detection of odorants which give rise to the smell. The ORNs generate odor-specific electrical signals (action potentials or pulses) in response to the binding of the odorants to the membranes [89, 15]. Activated ORNs trigger pulses, transmitting information about the odor to the processing centres in the insect brain. The *Drosophila* adult has about 1300 ORNs, whereas the *Drosophila* larva has 21 ORNs [90, 45]. Contrarily, the olfactory system of vertebrates has about thousands of ORNs [91]. The architecture of the olfactory pathway systems of the vertebrates are surprisingly identical to the one of the *Drosophila* [83]. Consequently, the olfaction of a *Drosophila* larva is one of the smallest well-studied olfactory pathway models.

Stage II - Pattern Separation
The ORNs encode qualitative, quantitative and spatio-temporal information about the odors, which are relayed to the neurons located in the antennal lobe (AL) [92]. The ORNs form afferent projections into the antennal lobe. There are two types of interneurons identified in the AL, namely the local interneurons (LNs) and the projection neurons (PNs) [93]. The antennal lobe is the primary olfactory association center, that consists of spherical bodies, called the *glomeruli*. Each ORN arborizes within its unique glomeruli, and thus there are 21 glomeruli. Within a glomerulus, the axon of one ORN synapses with the dendrites of its cognate LN and PN [93], summing up to a total of 21 LNs and 21 PNs. The sensory code takes a different representation in the AL, which are governed by the interactions happening inside the glomeruli. Only the projection neurons send out signals from the antennal lobe to the next brain region, i.e. the mushroom body. The transmission of the signals of the LNs remains within the AL. As the *Drosophila* larva only has 21 ORNs to encode a thousands of odors, many odors will elicit similar patterns of input activity. This is where pattern separation comes into play, which is the process that minimizes the overlap between patterns of neuronal activity with similar input profiles. A fitting analogy is to discern between the letter *A* and the same letter with an asterisk *A** [94]. Studies have shown that odors may activate the same subset of ORNs, which activates the same subset of glomeruli as the ORNs activate the cognate LNs and PNs homogeneously [95, 96]. As a consequence, pattern separation - moderated by local inhibition on the PNs - facilitates the ability to discern between odors with similar activation profiles. The PNs mirror the neural activity of their monosynaptic ORN input [20, 97, 98]. The same monosynaptic ORN input evokes the activity of the coupled, local interneurons [99]. The LNs, consequently, release the inhibitory neurotransmitter $\gamma-$aminobutyric acid (GABA), exerting an inhibitory control on the excitation of projection neurons [57]. This GABA-mediated inhibition influences particularly the activity of PNs in the discrimination of similar odorants, i.e. to amplify contrast between similar odor profiles [99, 57]. Each local interneuron forms an one-to-all connectivity with the projection

neurons, e.g. one LN transmits a widespread local inhibition to all PNs within the antennal lobe [93, 100]. Although, on a molecular level, there are different types of PNs and LNs [98, 20], governed by the type of ligand and the release of neurotransmitters, we will only focus on the most prominent features and mechanisms as explained above, without affecting its conclusions.

Stage III - Convergent encoding
The mushroom body (MB) receives the arborizations of the projection neurons and further transforms the sensory code [101]. The *Drosophila* adult has approximately 2200 Kenyon cells per hemisphere [49], and 72 in its larval phase [48]. The convergence from PNs to KCs employs a random connectivity, which is not subjected to plasticity or learning, but has an innate, taxon-specific connectivity [84]. Each KC typically has a dendritic branch with on average two to seven claws, which form random contacts to different PN>KC synapses [102, 103]. As a result, the overall response selectivity in the KCs reduces from 50% in the projection neurons to approximately 5% KC activation to a given odor [20, 21]. This is assumed to be the basis for the highly stimulus-specific response properties of the entire *Drosophila*'s olfactory pathway, resulting in a sparse spatio-temporal representation of an odor [92].
Each hemisphere in the *Drosophila* has a single GABAergic anterior paired (APL) [104]. The APL facilitates olfactory learning by forming a feedback inhibition with the Kenyon Cells [104]. The APL neuron forms a feedback loop with the KCs through a one-to-all connection. This is illustrated in Figure 2.8. This process is innate and therefore does not induce synaptic plasticity.

2.3.3. Olfactory learning and memory

The fruit fly - just like any other animal - can survive in a dynamic environment with familiar and new stimuli. The olfactory pathway of the fruit fly can continuously learn how to act when encountering new odors. Studies have demonstrated that clusters of neurons create a recurrent system within the mushroom body (MB). They are the substrates that are highly involved in odor learning and associative memory, referred to as MB compartments [105] (see Figure 2.9). This part is referred to as Stage IV in Figure 2.8 including the DANs, MBONs and MBINs in the mushroom body.

MB compartments
In a *Drosophila* mushroom body, 15 MB compartments were discovered, containing the arborizations of KCs, MBONs, and DANs [87]. The outputs of the MBONs are fed back via so-called mushroom body interneurons (MBINs) into the DANs. Each DAN of an MB compartment evaluates the reward or punishment signals received at a time, and subsequently translates the pattern of the KC activity to an MBON output. It results in skewing the behavior by modifying attraction or aversion [87]. The assignment of valence to a sensory stimulus is a form of adaptive mechanism, which is stored and recalled upon presentation of the same odor stimulus [49, 106]. The mechanistic operation of the MB-compartments is still under investigation, and thus computational and physiological studies are involved in gaining more insight into the underlying mechanism of biasing the outputs of the compartments, i.e. biasing the MBONs [87]. Figure 2.9 explains the motif that MBONs (- or +) convey a valence (-/+) to the dopaminergic neuron (+/-). Each MBON forms an excitatory connection to the MBIN in the same MB compartment.

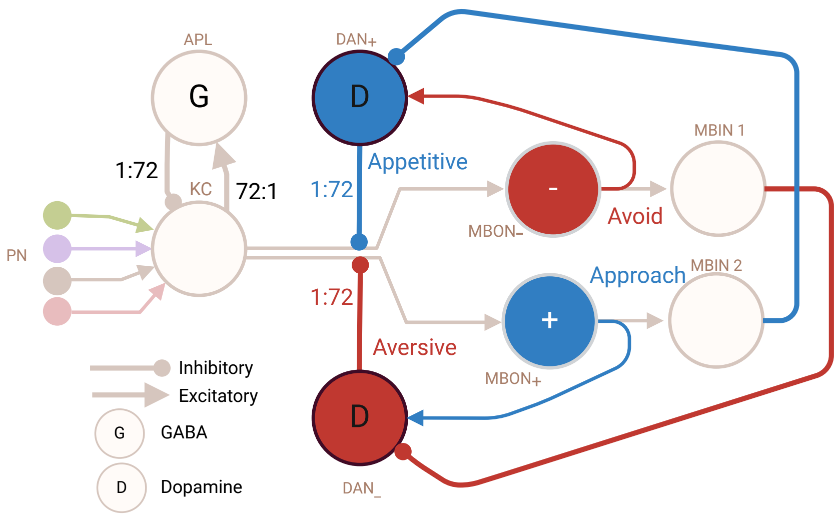


Figure 2.9: MB compartments Two MB compartments are shown in the figure, which is linked to either a positive valence (approach) or a negative valence (avoidance). Each MB compartment consists of an opposite-valued DAN which mediates in appetitive or aversive learning.

This MBIN inhibit the DAN in the opposite MB compartment [48]. There are additional modulatory neurons (MBIN), of which the identity of the neurotransmitter is unknown. This circuit motif, however, satisfies Dale's law [107], making it a plausible substrate. Dale's law postulates that a neuron cannot transmit both excitatory and inhibitory signals (i.e. a neuron releases the same type of neurotransmitters) [107]. For this reason, an intermediate neuron (MBIN) is necessary to provide the lateral inhibitory effect to the opposing MB compartment [48]. Consequently, each compartment is innervated by one MBON, one MBIN, and one opposite-valued DAN. Figure 2.9 graphically shows two MB compartments driving avoidance or approach through MBON- and MBON+, respectively. When MBON 1 is potentiated, it transmits excitatory signals to the opposite-valued DAN_+ in the same MB compartment. Dopamine is then released, which depresses the KC>MBON synapses. Simultaneously, through MBIN 1, DAN_- is inhibited. Therefore, the output of the mushroom body will be biased in favor of avoidance.

Dopaminergic facilitation

The fruit fly adapts its behavior by anticipating whether it should approach or avoid an odor source based on the "goodness" of the odor [87]. This decision-making process guides the selection of the action for any odor, which is mediated by dopaminergic neurons (DANs) [108]. The principle of dopaminergic facilitation in learning originates from conditioning paradigms. Conditioning is a physiological process - discovered by the Russian physiologist Ivan Pavlov - whereby an animal learns to associate a given stimulus with an action. It involves a form of reinforcement to drive the desired, behavioral response to the stimulus [109].

In a classical conditioning process, a stimulus with no intrinsic reward value will function as a rewarding stimulus after being repeatedly associated in time with a rewarding object. This rewarding object is called an unconditioned stimulus (US). After repetitive conditioning trials, the subject has learned to associate the rewarding object with an intrinsic reward value. Then, this stimulus is called a conditioned stimulus (CS) [110].

The generation of intrinsic values is modulated by the DANs in the mushroom body of the *Drosophila*, which effectively serves as a ”teacher” that corrects the animal’s behavior through neuromodulation [65]. The fruit fly learns to continuously make predictions in dynamic environments that favor them. These predictions are then paired with a specific behavioral action, which is then ”stored” in the synaptic response of DANs [88]. The underlying motif of dopamine neurons resembles the idea of prediction error coding [65] in reinforcement learning.

Prediction error

A prediction error is the difference between the predicted and the real received reward. The reward is either better than, equal to, or worse than the prediction. This prediction error influences future behavior, i.e. induces learning. If the reward deviates from its prediction, a prediction error exists and the prediction should be updated to modify the behavior. In the case of a positive prediction error, the actual reward is better than predicted, and thus the previous behavior must be adjusted to receive the reward in the future. On the other hand, if the real reward is worse than predicted (i.e., a negative error prediction), the behavior will be adjusted accordingly to avoid the wrongly-anticipated action in the future. Lastly, if the prediction error is zero, there is no difference between the prediction and the reward, and thus the behavior remains the same. In this case, there is no learning [111]. Figure 2.10 explains how dopamine release mediates in prediction error coding. When the received outcome differs from the prediction, an error, either negative or positive, arises. The valence of this prediction error is encoded by dopamine release, which updates the prior prediction. Dopamine plays a crucial role in computing prediction errors and facilitating them through top-down signaling. It is a teaching signal that affects the synaptic plasticity in associative learning in the *Drosophila* [111].

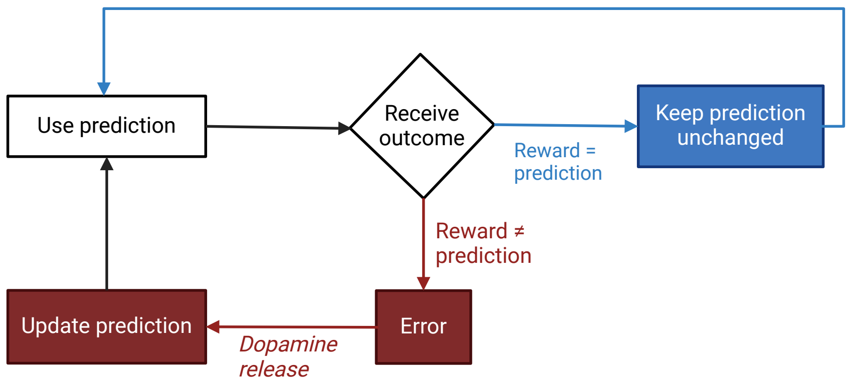


Figure 2.10: Prediction error coding using dopamine Dopamine is used to update a prediction for a given stimulus. When the same stimulus is presented, the model will use the same prediction. If the model receives a reward for the anticipated prediction, the prediction remains unchanged and the prediction is zero. If the reward is never received, then a prediction error is generated. Dopamine will be released as a result of a prediction error. The figure is inspired by Schultz et al. [111].

2.4. State-of-the-art hardware solutions

We will have a look at the state-of-the-art solutions of olfactory hardware implementations with a focus on neuromorphic designs. Neuromorphic hardware significantly differs from hardware with a Von Neumann architecture. Figure 2.11A shows a typical Von Neumann architecture where the computing resources (CPU/GPU) are separated from the memory [112]. A spiking neural network that runs on a Von Neumann computer constantly exchanges data back and forth, which creates a bottleneck in applications where high throughput and speed are necessary [113]. Therefore, neuromorphic computing proposes a novel architecture that is inspired by a network of biological neurons, where computing and memory are co-located. As seen in Figure 2.11B, one core constitutes one neuron, multiple synapses, and a communication block. In this way, data transmission is done in a more distributed and parallel way [113]. A spiking neural network is built by connecting multiple (neural) cores. In neuromorphic hardware architectures, the address-event representation (AER) scheme is the de facto communication protocol. It encodes the time of spike and address of the firing neuron in a data package which is routed to other neurons in the connectivity scheme [114].

Generally, neuromorphic designs using spiking neural networks can be categorized into two classes (i) large-scale neural networks focused on high computing performance and parallelism, and (ii) small-scale networks exploiting biophysical principles and targeting biologically realistic modeling. The SpiNNaker [116] , TrueNorth [117], and Intel's Loihi [40] chips belong to the former. In olfactory studies, large-scale networks have been recreated in software and hardware, which targeted the mammalian olfactory bulb [42, 31, 39, 43, 41]. In these studies, underlying principles such as feedback inhibition, reinforcement, or neuromodulation have been excluded. On the other hand, other studies focused on closely reconstructing the olfactory pathway model of the *Drososophila* larva via software simulations using for example the Brian2 simulator [29, 32]. The Brian 2 simulator [118] is a neural simulator written in Python that provides a platform to simulate a single neuron or a network of neurons in detail. The work from Jürgensen et al. in [29] utilizes the olfactory pathway model as described in Figure 2.8, which is an elaborate software model targeted on simulating this model. Only a few studies have implemented this model to explore its efficacy in hardware [36, 44, 33], which is the focus of the thesis. In the following, we will cover the main hardware designs on which odor classification has been benchmarked.

DYNAPs

DYNAPs is a chip with one thousand VLSI neurons, distributed among four neural cores with an address-event representation (AER) bus for data transmission [115]. Each core has 256 neurons based on an LIF neuron model [119]. Furthermore, the synapses are modeled by producing inhibitory and excitatory post-synaptic currents (IPSC, EPSC) with time constants ranging from microseconds to milliseconds [120, 119]. This chip employs an in-memory computing architecture thanks to the distributed memory elements (SRAM and capacitors). This is different from other fully digital designs, that time-multiplex (i.e., having a shared module for updating logic and a central storage unit [10]) the computing resources and still face the transmission of state memory from different computing areas to memory storage, e.g. the TrueNorth architecture [113]. Figure 2.11C shows the architecture of one DYNAPs node, i.e. one neuron. The figure shows that the neuron

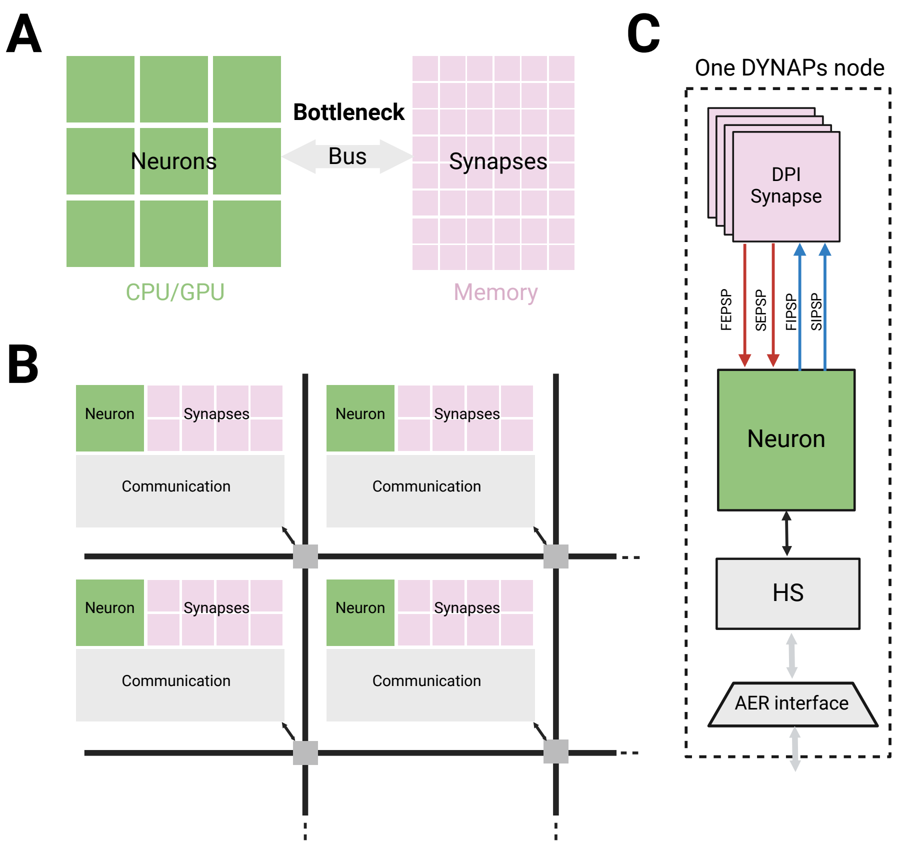


Figure 2.11: Von-Neumann architecture and (neuromorphic) non-Von-Neumann, distributed architecture (A) A typical Von Neumann architecture with the computing blocks separated from the memory [113] (B) A neuromorphic hardware architecture with neural cores consisting of one neuron, synapses, and a communication block [113] (C) An example of a neuromorphic, analog architecture for a neuron. It shows one DYNAPs computing node or neuron that is connected to four DPI circuits emulating synapses. The neuron checks if the membrane potential has been exceeded. Then, this information is encoded in an AER package in the AER interface, after passing through a handshaking (HS) block [115].

and the synapses are clustered in the same processing element, merging computing and storage. The state of the neuron is packaged in the handshaking block using the AER protocol, after which it is distributed with other neurons via the AER interface bus.
This implementation has been successfully used in demonstrating odor specificity and similarity, for three artificial odors with an overlapping odor profile. The model uses a similar model as shown in Figure 2.8, excluding the Stage IV neurons (DANs, MBONs, and MBINs). This implementation deploys a small, non-plastic larval network that demonstrated similarities with (1) inhibitory feedback from the APL through inhibitory currents and (2) short-term effects with SFA. However, the biasing of the currents encountered instability issues specifically for SFA. An enhanced version of DYNAPs, DYNAP-SE2, was introduced at the start of 2024 [121]. It has a dedicated block for spike frequency adaptation that may improve the process of biasing. Furthermore, this chip includes adaptation on a longer timescale through a homeostasis mechanism. On this chip, this synaptic scaling ensures that the excitability stays within a target range. However,

DYNAP-SE2 has not been used for the modeling of the olfactory network.

Intel Loihi and Lava framework
In 2018, Intel's Neuromorphic Research Lab introduced the Loihi neuromorphic chip with a fully digital architecture [40]. A Loihi chip integrates 128 neuromorphic cores, each comprising 1,024 primitive spiking neural units grouped into sets of trees, forming neurons. These cores feature LIF neuron models, utilizing forward Euler. The approximations of the LIF neuron model are modeled be constant voltage decay factor $\lambda \in 0, 1$, which accounts for leakage and is typically pre-trained [122]. All parameters and values in this chip use fixed-precision format. It accommodates any weight precision ranging from one to nine bits, whether signed or unsigned. Notably, all signals in the system are digital, and the networks operate as discrete-time dynamical systems.

For instance, Imam et al. demonstrated lifelong learning by identifying odors from high-dimensional noisy olfactory signals using chemosensor arrays [123], leveraging other biologically inspired mechanisms implemented on Intel's Loihi [39]. This model is inspired by the mammalian olfactory bulb, comprising at least 10,000 neurons. The algorithm's high performance is attributed to various mechanisms, including neuromodulatory optimization of circuit properties and local learning rules. Additionally, sparse excitatory and dense inhibitory networks were utilized to temporally encode information. An inference cycle - modelling a ”sniff” - required 2.75 ms and consumed 0.43 mJ of energy, of which 0.12 mJ is dynamic energy [39].

The new Loihi 2 chip has programmable neurons, which are described by microcode instructions [122]. Furthermore, Loihi 2 supports up to 1 million neurons and 123 million synapses. Its enhanced external connectivity includes GPIO and SPI communication buses. Intel offers access to Loihi (1 and) 2 hardware via the Neuromorphic Research Cloud, which is a cluster of virtual machines (VMs) and Loihi systems that can be accessed from an SSH terminal. These can be used to develop and benchmark custom neuromorphic algorithms and applications. Furthermore, for prototyping neural networks, Intel launched an open-source Python-based framework for neuromorphic applications, called Lava [124]. Lava offers a software platform to develop and evaluate spiking neural networks for Loihi 2, with many libraries describing software models of neurons. Only models which are also programmed in microcode for the virtual Loihi 2 can be used in tests on the Loihi 2. As it is an open-source framework, the platform is constantly expanding and evolving. It offers distinct advantages over other software frameworks, such as Brian 2, PyNN, PyTorch, and TensorFlow, in neuromorphic computing. SNNs in Lava can either run on GPU or on a cloud-based version of Loihi 2, thereby supporting the option to run them on a neuromorphic platform. Lava facilitates mapping neural networks onto hardware platforms, circumventing the design overhead of transitioning from the software model to its hardware counterpart. Additionally, Lava provides a low-level interface for mapping neural networks onto neuromorphic hardware, facilitating a seamless integration. Lava possesses characteristics such as modularity, and software-to-hardware extensibility. Appendix A elaborates on setting up a neural network using Lava and explains how new neuron models can be created within the framework. Lava and Loihi 2 have not tyet been explicitly used in modelling any type of olfactory pathway modelling.

FPGAs

Field-programmable gate arrays (FPGAs) provide a hardware platform which is more reconfigurable and flexible than analog and mixed-signal designs. FPGAs can also be used in the early stages of hardware development for prototyping and testing new designs before committing to chips [10]. A neuromorphic implementation of the mammalian olfactory bulb was implemented with a Virtex II pro FPGA [41]. The model was tested on a two-odor classification task, whereby the synaptic weights were determined during a training stage. It was used to classify learnt patterns representing distinct odor stimuli. The neurons were modeled as integrate-and-fire (IF) units, without the leaky part. The synapses use a 16-bit representation, while the soma uses a 32-bit representation. When a spike occurs within a clock cycle, the total current is incremented by a fixed value. Although providing one of the few FPGA-based modelling of olfactory pathways, several simplifications were made (such as fixed time constants and a resting potential of 0 V [41, 125]), which restricts flexibility and versatility. Moreover, since the weights were pre-trained values, a critical, computational primitive in bio-inspired models as to (unsupervised) and dynamic synaptic weight modifications is missing.

3 Design and Implementation

This chapter presents the proposed design and provides insights into translating the concepts into its software and hardware implementation. The first section outlines the proposed model of the *Drosophila* larva to explain the computational framework and modeling methodologies. The remaining part of this chapter consists of two sections: (i) software and (ii) hardware design.

3.1. Proposed model

The conceptual framework for the larval olfactory pathway in *Drosophila*, as proposed by Jürgensen et al. in [29], serves as the foundation for the model presented herein. The architecture of this model was presented in Section 2.3, depicted in Figure 2.8. Jürgensen et al.'s work highlighted the role of dopaminergic neurons in facilitating olfactory learning through the reinforcement motif, driven by prediction error coding. In order to facilitate this learning work, a conductance-based LIF (COBALIF) neuron is developed that can capture the key spatio-temporal processes - such as sparsity, plasticity (including SFA), and neuromodulation - on hardware. The original model was developed in the Brian 2 simulator [118], which runs on a CPU/GPU. To this end, a neuromorphic-oriented methodology was adopted, leveraging Intel's Lava (Python-based) framework and creating an FPGA design. Consequently, the initial model underwent reconstruction to facilitate translation from software to neuromorphic hardware. This section, first, introduces the mathematical model, delineating the computational processes governing neuron behavior of the COBALIF neuron, membrane potential updates, inhibition, and spike frequency adaptation.

3.1.1. Computational model

The computational model of the proposed system is described by the first-order ODE Equation 2.6. This neuron type utilizes conductance-based synapses to account for the ionic streams but lacks a component to model SFA, as explained in Section 2.1. The reconstruction of SFA in this model is enabled through an additional adaptation current [29] that is expressed in the same conductance-based (g_{Ia}) fashion as the other components (i.e. excitatory, inhibitory, and leakage current). In the olfactory pathway model of the *Drosophila* larva, SFA has not been identified for all neuron types, hence the inclusion of SFA to a neuron's equation only applies to the ORNs, KCs, DANs, MBONs, and MBINs [29]. The ODE of Equation 2.6 is extended with a fourth component:

$$C_m \frac{dV}{dt} = g_L(E_L - V(t)) + g_e(E_e - V(t)) - g_i(E_i - V(t)) - g_{Ia}(E_{Ia} - V(t)) \tag{3.1}$$

where E_{Ia} is the adaptation resting potential.

3.1.2. COBALIF neuron

To approximate the behavior of an ODE, we apply a numerical integration method, i.e. forward Euler. The application of the Euler method in computational models is widely used. Previous hardware implementations primarily focus on approximating temporal varying components through parametrization and tuning techniques of temporally varying components [42, 125, 41]. However, for the proposed model, the Forward Euler method is applied to maintain the integrity of these time-varying components within the computational model. Forward Euler can be applied to numerically solve this ODE, because the initial condition is given. This is related to the fact that the membrane potential is equal to its resting potential V_r at $t = 0$.

Prior to applying forward Euler, Equation 3.1 is rewritten by exposing the $V(t)$:

$$\frac{C_m}{g_L + g_e(t) - g_i(t) - g_{Ia}(t)} \frac{dV(t)}{dt} = -V(t) + \frac{g_L E_L + g_e(t)E_e - g_i(t)E_i - g_{Ia}(t)E_{Ia}}{g_L + g_e(t) - g_i(t) - g_{Ia}(t)} \tag{3.2}$$

After substituting the forward finite difference of the ODE in Equation 3.2, we arrive at the following equation:

$$\frac{C_m}{g_L + g_e(t) - g_i[t]) - g_{Ia}[t]} \frac{V[t + \Delta t] - V[t]}{\Delta t} \approx -V[t] + \frac{\overbrace{g_L E_L + g_e[t]E_e - g_i[t]E_i - g_{Ia}[t]E_{Ia}}^{I_{ss}[t]}}{g_L + g_e[t] - g_i[t] - g_{Ia}[t]} \tag{3.3}$$

Equation 3.3 represents the approximated solution to the ODE. The goal of solving this equation is to compute the membrane potential that is governed by time-varying input currents (excitatory, inhibitory, adaptation, and leakage). The time step Δt must be sufficiently small, such that we can assume that these input currents can considered to be constant during this time interval. Therefore, the numerator on the right-hand side of Equation 3.3 - representing the total neuron current - is assumed to be constant during this time interval, which we refer to as the steady-state current I_{ss}. Equation 3.3 is rewritten to only have the difference on the left-hand side:

$$V[t + \Delta t] - V[t] = -V[t] \cdot \frac{\Delta t}{\left[\frac{C_m}{g_L + g_e[t] - g_i[t] - g_{Ia}[t]}\right]} + \frac{I_{ss}[t] \cdot \Delta t}{C_m} \tag{3.4}$$

Equation 3.4 is rewritten to yield the solution of the membrane potential ($V[t + \Delta t]$):

$$V[t + \Delta t] = V[t] - V[t] \cdot \frac{\Delta t}{\tau[t]} + \frac{I_{ss}[t] \ \cdot \Delta t}{C_m} \tag{3.5}$$

where $\tau(t)$ is the (time-varying) time constant that is expressed in brackets in Equation 3.4. Equation 3.5 can be rewritten for better readability to arrive at the final solution of the

ODE. The (final) time-stepped solution $V[t+\Delta t]$ is for each simulation time t:

$$V[t+\Delta t] = V[t](1 - \underbrace{\frac{\Delta t}{\tau[t]}}_{dv[t]}) + \frac{I_{ss}[t] \cdot \Delta t}{C_m} \tag{3.6}$$

Equation 3.6 is dependent on two time-varying variables, namely the neuron-dependent time constant $\tau[t]$ and the steady-state current $I_{ss}[t]$ within a time interval. Both variables are dependent on the time-varying excitatory, inhibitory, and adaptive conductance values: $g_e(t)$, $g_i(t)$ and $g_{Ia}(t)$, respectively. As explained in Section 2.2, the conductances are exponentially decaying, governed by their respective time constants $\tau_e(t)$, $\tau_i(t)$ and $\tau_{Ia}(t)$. Hence, each conductance is expressed by a linear ODE with a solution in the form of $g(t) = g_0 e^a$, whereby a is a time-varying parameter. The real expression of the linear ODEs of conductances is, where subscripts indicate the nature of the conductance :

$$\frac{dg_{i,e,Ia}(t)}{dt} = -\frac{g_{i,e,Ia}(t)}{\tau_{i,e,Ia}(t)} \tag{3.7}$$

Forward Euler is again applied to this equation to arrive at the solution:

$$g_{i,e,Ia}[t+\Delta t] = g_{i,e,Ia}[t] e^{-\frac{\Delta t}{\tau_{i,e,Ia}[t]}} \tag{3.8}$$

Since this equation is a linear equation with exponential decay, whereby the timestep (Δt) is 0.1 ms (as denoted in [29]), and the time constants are in the order of a few to hundreds of milliseconds, it can be approximated by:

$$g_{i,e,Ia}[t+\Delta t] = g_{i,e,Ia}[t] \cdot (1 - \frac{\Delta t}{\tau_{e,i,Ia}[t]}) \tag{3.9}$$

The synaptic conductances are computed after each spike according to Equation 3.9. Then, the steady-state current $I_{ss}[t]$ is calculated for each neuron in a population:

$$I_{ss}[t] = g_L E_L + g_e[t] E_e - g_i[t] E_i - g_{Ia}[t] E_{Ia} \tag{3.10}$$

Furthermore, the change in membrane potential for each neuron is expressed as $dv[t]$, which appeared in Equation 3.6:

$$dv[t] = \frac{\Delta t}{\tau[t]} = [g_L + g_e[t] - g_i[t] - g_{Ia}[t]] \frac{\Delta t}{C_m} \tag{3.11}$$

whereby the last component is a constant, containing the chosen simulation time step ($\Delta t = 0.1$ms) and the neuron-specific membrane capacitance C_m.

The equations for $I_{ss}[t]$ (Equation 3.10) and $dv[t]$ (Equation 3.11) provide the transient dynamics of biological principles. As explained in Section 2.3, each neuron type may have different parameters, which are related to the construction of the ODE. Table 3.1 outlines the equations for all neurons, which will be addressed in the implementation. The overview thus excludes the equations for the MBONs, MBINs, and DANs. Each equation follows the same computational steps as explained above.

Table 3.1: The equations of the ORN, PN, LN, KC, and APL neuron in the sub-threshold regime, expressing the membrane potential over time for a neuron n in the population.

Neuron Type	Sub-threshold ODE
ORN	$C_m^{ORN}\frac{d}{dt}V_n^{ORN}(t) = g_L^{ORN}(E_L^{ORN} - V_n^{ORN}(t)) + g_e^{InputORN}(E_e - v_n^{ORN}) - g_{Ia}^{ORN}(E_{Ia} - V_n^{ORN}(t))$
PN	$C_m^{PN}\frac{d}{dt}V_n^{PN}(t) = g_L^{PN}(E_L^{PN} - V_n^{PN}(t)) + g_e^{ORNPN}(E_e - V_n^{PN}) - g_i^{LNPN}(E_i - V_n^{PN}(t))$
LN	$C_m^{LN}\frac{d}{dt}V_n^{LN}(t) = g_L^{LN}(E_L^{LN} - V_n^{LN}(t)) + g_e^{ORNLN}(E_e - V_n^{LN}(t))$
KC	$C_m^{KC}\frac{d}{dt}V_n^{KC}(t) = g_L^{KC}(E_L^{KC} - V_n^{KC}(t)) + g_e^{PNKC}(E_e - V_n^{KC}) - g_i^{APLKC}(E_i - V_n^{KC}(t)) - g_{Ia}^{KC}(E_{Ia} - V_n^{KC}(t))$
APL	$C_m^{APL}\frac{d}{dt}V_n^{APL}(t) = g_L^{APL}(E_L^{APL} - V_n^{APL}(t)) + g_e^{KCAPL}(E_e - V_n^{APL}(t))$

3.2. Software design

The software model is developed utilizing Intel's Lava framework. This section is focused on the software design of the COBALIF neuron in the Lava framework. Firstly, it discusses how synapses are modeled. Then, we will explain how the COBALIF neuron is implemented in Lava using a custom algorithm.

3.2.1. Synapse models

In Lava [124] and in Brian 2 [34], the synaptic connections between neurons are instantiated through a connectivity matrix $C_{M\times N} \in \mathbb{Z}$, where N, M denote the number of pre-synaptic and post-synaptic neurons, respectively. Each element in the connectivity matrix is binary, taking values of either 0 or 1 to indicate the presence or absence of a synapse. Figure 3.1 shows the connectivity matrix of the PNs and KCs. The number of pre-synaptic neurons (N) is equivalent to the number of PNs, i.e., 21. These form a connection through the connectivity matrix with 72 KCs, which are post-synaptic neurons (M). Each line in the figure expresses a flagged entry with the value 1 in the connectivity matrix. In ANNs, the value of the neuron (y_M) follows from an element-wise matrix multiplication with a weight matrix ($W_{M\times N} \in \mathbb{R}$) and the pre-synaptic spike matrix (i.e., input vector $x_N \in \{0, 1\}$) at a specific time. This is then passed through an activation function. In contrast, modeling conductance-based spiking neurons in Brian 2 and Lava, are governed by an extensive mathematical model. Hence, in this case, the state of the neuron's value follows from

$$y_M = f(x_N^{\mathsf{T}} \times W_{N\times M} \times C_{M\times N}) \tag{3.12}$$

where f refers to the chosen mathematical function, which in our case is related to the solution of $V[t + \Delta t]$ in Equation 3.6. In Lava, this connectivity matrix is instantiated through a construct `ProcessModel`, which is explained in detail in Appendix A.

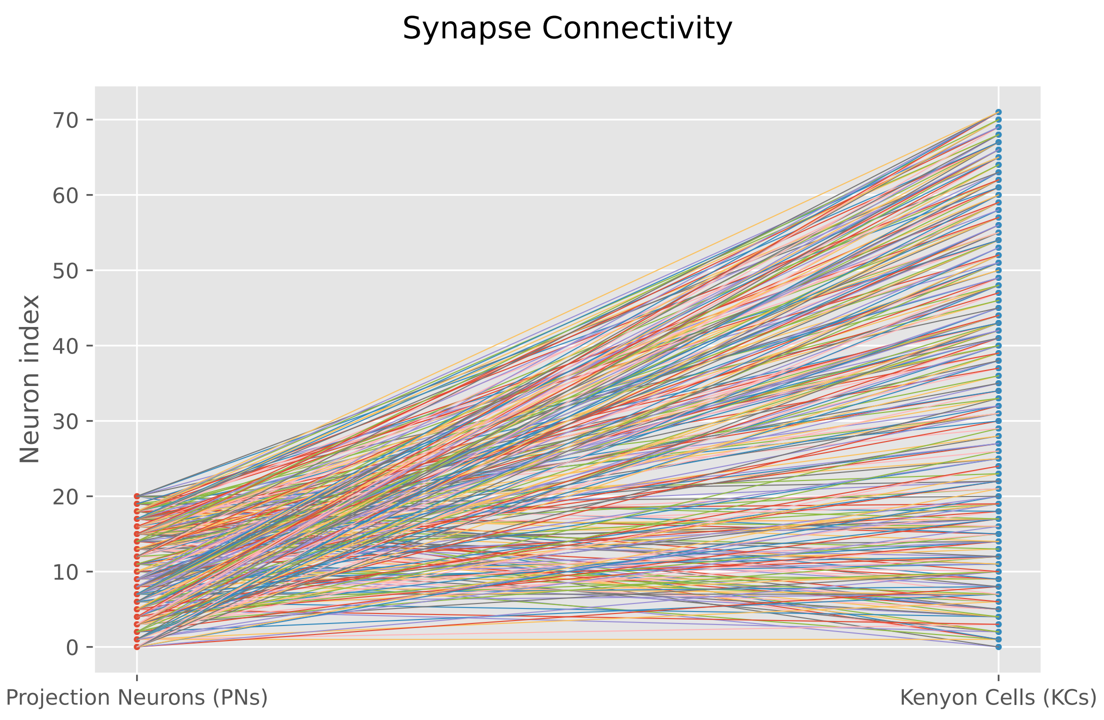


Figure 3.1: Synapse connectivity graph between PNs and KCs governed by the connectivity matrix $C_{M \times N}$. The projection neurons ($N = 21$) are the pre-synaptic neurons that are randomly connected to 2 to 7 KCs ($M = 72$).

3.2.2. COBALIF neuron in Lava

Leveraging the approximations of the COBALIF neuron from Section 2.2, an algorithm is developed (Algorithm 1), encompassing four distinct steps: (1) Spike checker, (2) Computation of $V[t + \Delta t]$, (3) Computation of the conductances, and (4) Comparator. This algorithm describes the steps, being executed within one timestep, when the neuron is not in its refractory period τ_{r}. The software model updates a register of the simulation times of each neuron in a matrix T. The neuron in the population is indicated by n. If the neuron spiked in a previous timestep, the refractory period (τ_r) is added to the current simulation time. This is saved in a register of the time during which the neuron is in refractory T^{refrac}. The neuron in the sub-threshold regime adheres to the following functional steps:

1. **Spike checker**: If the neuron (n) is not in the refractory period (i.e. $T \leq T^{refac}$), it will be passed through this step. Within each timestep, the code examines whether excitatory and/or inhibitory pre-synaptic spikes have been received and adjusts the respective synaptic strengths accordingly.
2. **Computation of** $V[t + \Delta t]$: This stage involves the computation of the membrane potential for the subsequent timestep, according to Equation 3.6. This computation necessitates the determination of steady-state current $I_{ss}[t]$ (Equation 3.10) and potential difference $dv[t]$ (Equation 3.11) at the current simulation time t.
3. **Computation of the conductances**: Conductance values are updated according to the neuron's dynamics according to Equation 3.9.

4. **Comparator**: In this phase, the algorithm evaluates whether the neuron's membrane potential at [t + dt] surpasses the threshold. If this condition is met, a spike is generated and transmitted through the output port. Otherwise, the spike value remains zero. Additionally, depending on whether the neuron undergoes SFA, a constant value γ is added to the adaptation conductance, thereby augmenting the adaptation current. If the neuron spiked during the timestep, the respective membrane potential will return to its resting potential V_r. Furthermore, the output spike of the respective neuron (S^{out}) will be given a (timestamped) spike.

This algorithm is instantiated using the structure of Lava's `ProcessModel`. As described in Table 3.1, the sub-threshold dynamics of neurons within a layer adhere to identical equations, with the constituent components specifying the nature of the conductance. Furthermore, the unique parameters of the neurons can be found in Appendix B.

Algorithm 1: COBALIF in sub-threshold dynamics

```
Input: $\mathbf{S}^e, \mathbf{S}^i$
Internal Variables   : $\mathbf{T}, \mathbf{T}^{refrac} \mathbf{g}_e, \mathbf{g}_i, \mathbf{g}_{Ia}, \gamma, \mathbf{V}_r, \mathbf{t}$
Result: $\mathbf{S}^{out}$

Step 1: Spike checker
while $T \leq T^{refrac}$ do
    for $n \in T$ do
        if $\left\langle S^e(n), S^i(n) \right\rangle = 1$ then
            $g_e(n) \leftarrow g_e(n) + w_e$
            $g_i(n) \leftarrow g_i(n) - w_i$
        else
            if $\left\langle S^e(n), S^i(n) \right\rangle = 0$ then
                $g_e(n) \leftarrow g_e(n)$
                $g_i(n) \leftarrow g_i(n)$
            end
        end
    end
    $g_{Ia}(n) \leftarrow g_{Ia}(n)$
end
Step 2: Compute $v^{n \in T}$ according to Eq. 3.6

Step 3: Compute $\langle g_i^{n \in T}, g_e^{n \in T}, g_{Ia}^{n \in T} \rangle$ according to Eq. 3.9

Step 4: Check if $v^{n \in T} \geq v_{th}$
if $v^{n \in T} > v_{th}$ then
    $\mathbf{g}_{Ia}(n) \leftarrow \mathbf{g}_{Ia}(n) - \gamma$
    $\mathbf{S}^{out}(n) \leftarrow 1$
    $\mathbf{V}(n) \leftarrow \mathbf{V}_r$
    $\mathbf{T}(n) \leftarrow \mathbf{T}(n) + \tau$
else
    $\mathbf{S}^{out}(n) \leftarrow 0$
end
```

3.3. Hardware design

The hardware implementation of the COBALIF neuron model is deployed on a Zynq 7000 SoC, using Verilog as the hardware description language. This hardware implementation leverages the groundwork laid by the software implementation of the olfactory pathway model, developed and simulated in Lava. The software model in Lava allows us to demonstrate the feasibility and the working operation of the COBALIF neuron at a neuron and network level. The FPGA-based hardware implementation provides a platform for replicating the dynamics inherent in the olfactory pathway model. To this end, the design concentrates on the realization of a single COBALIF neuron. This section is divided into two parts: (i) the digital datapath of the COBALIF neuron, and (ii) its digital control.

3.3.1. Digital datapath

This subsection specifically covers the digital path for the COBALIF neuron. It starts with the design choices regarding timing and the precision format, representing the numerical values in the digital design. Afterward, the proposed digital design for the COBALIF neuron is described, whereby the various design modules are presented.

Timing

The clock speed is 100 MHz and utilizes a synchronous clock. This design combines the features of timestepped and event-driven dynamics [10]. To reduce the data movement, we designed controllers to gate data movement between registers and solver modules during refractory states. However, whenever an event (i.e., a spike) occurs, the systems follow an updating scheme moderated by counters and controllers, which is unique to the neuron. This requires each neuron to have two counters (their main counter and down counter), which accommodate event-driven updates.

Precision format

In neuromorphic hardware implementations, the choice between single floating-point precision and fixed-point precision entails a trade-off between computational accuracy and resource efficiency. Single floating-point precision offers higher accuracy by allowing for the representation of a wider range of values and finer granularity in numerical computations [126]. This precision is particularly beneficial in applications requiring precise modeling of neural dynamics and synaptic weights. However, the use of single floating-point precision necessitates greater hardware resources, including increased memory and computational units, resulting in higher power consumption and area footprint [127].

On the other hand, fixed-point precision offers a more resource-efficient alternative by representing numbers with a fixed number of integer and fractional bits [126]. While fixed-point precision consumes fewer hardware resources compared to single floating-point precision, it may result in reduced numerical accuracy and dynamic range, potentially leading to quantization errors and loss of precision in neural computations [127]. Consequently, the choice of the precision format is decided upon preserving the accuracy of the computational values and thus having a larger dynamic range, which ranges in magnitudes from nano to milli. Therefore, this design utilizes a single-precision floating-point format.

The Xilinx Floating Point Operator (v7.1) [128] is a versatile hardware module designed to perform floating-point arithmetic operations efficiently within FPGA-based systems. The Xilinx Floating Point Operator integrates with the AXI-4 (Advanced eXtensible Interface 4) protocol [129], which is a widely adopted standard for interconnecting and communication between intellectual property (IP) cores within FPGA-based systems [129]. Figure 3.2 shows the core schematic symbol including its input operands, output, and control signals. This operator has two input ports for the data of operands A and B, which are processed in the core to generate the data for the output. The arithmetic operation needs to be selected before utilizing it in a design module. The control and the respective protocols are discussed in Section 3.3.2.

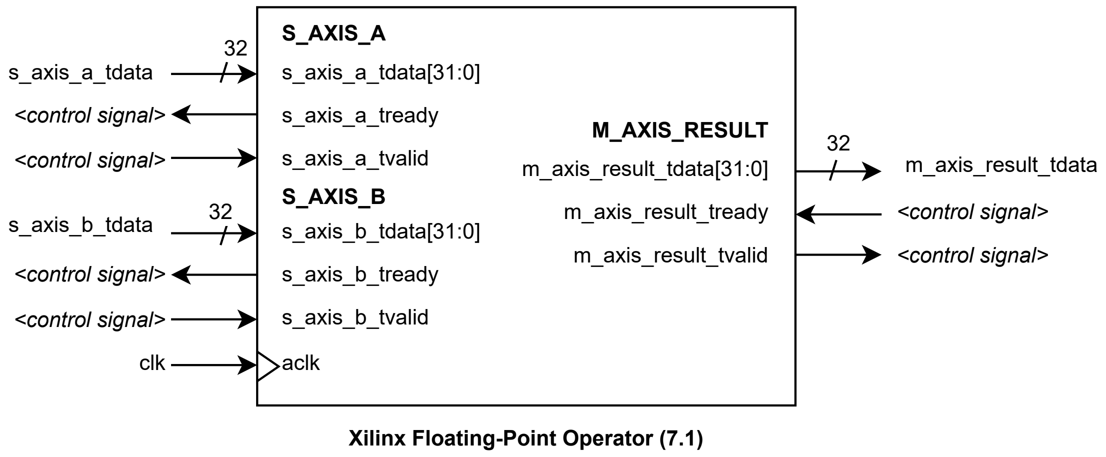


Figure 3.2: Xilinx Floating-Point operator (V7.1) core schematic symbol [128]. The Floating-Point operator is divided into two input clusters (`S_AXIS_A` and `S_AXIS_B`) and one output cluster (`M_AXIS_RESULT`), and follows the AXI4-Stream porting protocol. The directions of the arrows indicate whether the ports are input or output ports. Each cluster contains 32-bit operand input (using single-precision floating-point format), which are denoted by `s_axis_a_tdata` and `s_axis_b_tdata` for operand A and B, respectively. The output operand is given by `m_axis_result_tdata` (32 bits). All control signals are denoted by `<control signal>` and are enabled with an active high. There are input control signals (`s_axis_a_tvalid`, `s_axis_b_tvalid`, `m_axis_result_tready`) and output control signals (`s_axis_b_tready`, `s_axis_a_tready`, `m_axis_result_tvalid`). All signals are synchronous to the `aclk` input, which is the system clock (`clk`) in our design.

Digital neuron unit design

The digital design of the COBALIF neuron adheres to Algorithm 1 in the software model. The digital architecture of a single digital neuron unit (DNU), as depicted in Figure 3.3, consists of multiple modules:

- **Refractory timer**: This module monitors the spiking activity of the neuron at each time step and emits a signal (`REFRACTORY`) indicating the neuron's refractory state. This signal is essential to prevent arithmetic modules from performing computations during the refractory period, accounting for a reduction in computations. Internally, a countdown mechanism is employed to track the passage of the refractory period τ_r, which is set to a duration of 2 milliseconds (see Appendix B [29]). The counter is activated if a spike (`SPIKE`) is generated and if `MAIN_COUNT` is equivalent to zero, ensuring that the downcounter counts down in the following timestep. Upon completion of the refractory period, the refractory signal is de-asserted, allowing the computing modules to resume operation.

- **Conductance update and storage**: The module is managed by two control signals, SPIKE and EVALUATE. The updates on the excitatory, inhibitory, and adaptation conductance ($g_e, g_i,$ andg_{Ia}) are triggered when EVALUATE is active high, as specified in Step 3 of Algorithm 1. If the neuron spiked in the same timestep and has SFA, the SPIKE signal triggers the augmentation of g_{Ia} with a constant SFA value. Subsequently, the module stores these updated values for utilization in the next timestep.
- **Membrane update and storage**: Within this module, the COBALIF algorithm (Step 2 in Algorithm 1) is executed, provided that the neuron is not in a refractory state. This happens in a sequential whereby firstly EVALUATE triggers the computations of $I_{ss}[t]$ and $dv[t]$. Subsequently, after de-asserting this signal, COMPUTE triggers the computation of $V[t + \Delta t]$.
- **Voltage comparator**: $V[t + \Delta t]$ undergoes comparison with the threshold voltage (V_{th}) within the same timestep, triggered by the ASSESS signal. In the event of a neuron spike, an active SPIKE signal is generated, which is relayed to other modules to initiate the refractory period.
- **MUX 1 and MUX 2**: MUX 1 operates under the control of the MODE 0, 1 signal, determining the selection between the available synapse weight values for subsequent summation to the MUX. MUX 2 facilitates the selection of the voltage for the next timestep, which is controlled by the SPIKE signal. Only in case the SPIKE signal is asserted, MUX 2 will select the resting potential V_r. Otherwise, the next value for the membrane potential is the one computed in the same timestep.

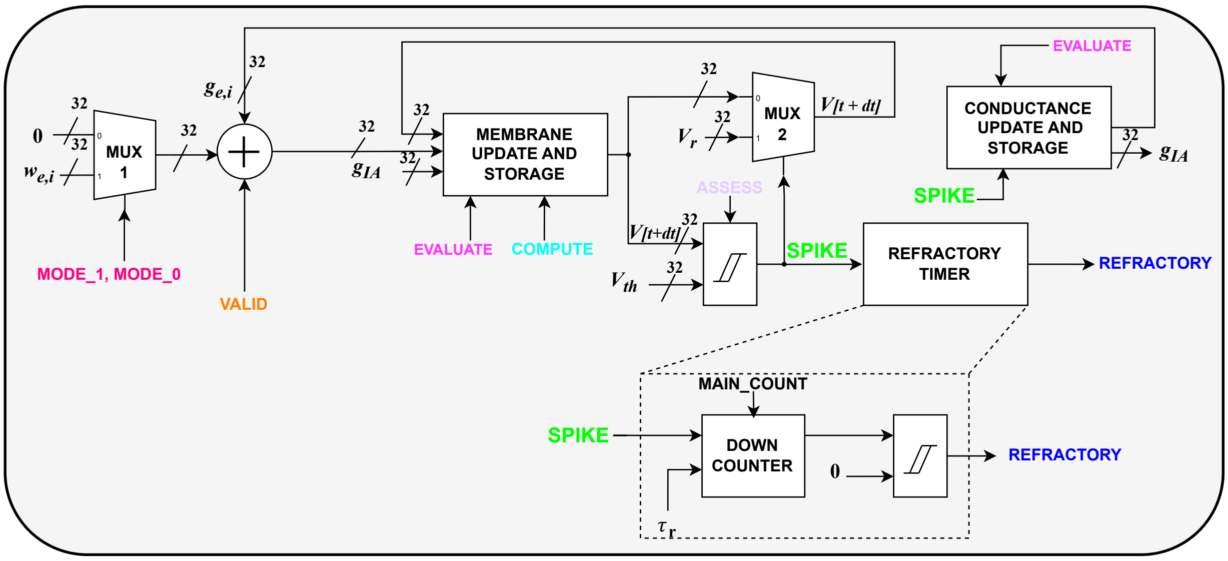


Figure 3.3: Architecture of a single digital neuron unit (DGU). This DGU consists of three main modules, which are the *Membrane Update and Storage*, *Conductance Update and Storage*, and the *Refractory Timer*. Controllers - not depicted in the figure - generate the control signals. The signals adhere to a color-coding convention to enhance clarity.

The *Membrane Update and Storage* and the *Conductance Update and Storage* consist of multiple arithmetic operators, which are integral to the computation of the membrane potential in the sub-threshold regime, like additions, subtractions, and multiplications. All arithmetic operations are covered by the Floating-Point Operator (v7.1) IP. The quantity of operations undertaken in Figure 3.4 provides an in-depth overview of these modules, detailing the interfacing of computations facilitated by the Floating-Point Operator. The *Membrane Update and Storage* encompasses computations for $I_{ss}[t]$, $dv[t]$, and $v[t + dt]$. Notably, the *ISS Module* omits the excitatory component, as the multiplication between the excitatory potential ($E_e = 0$ [80]) and the excitatory conductance yields zero (see Equation 3.10). This exclusion of the excitatory component within the *ISS Module* mitigates resource utilization while preserving computational efficiency. The *Conductance Update and Storage* consists of two sub-modules for (i) updating the excitatory and inhibitory conductances, and (ii) the SFA module for synaptic weight adaptation. Depending on the neuron type, these sub-modules may not need to be instantiated.

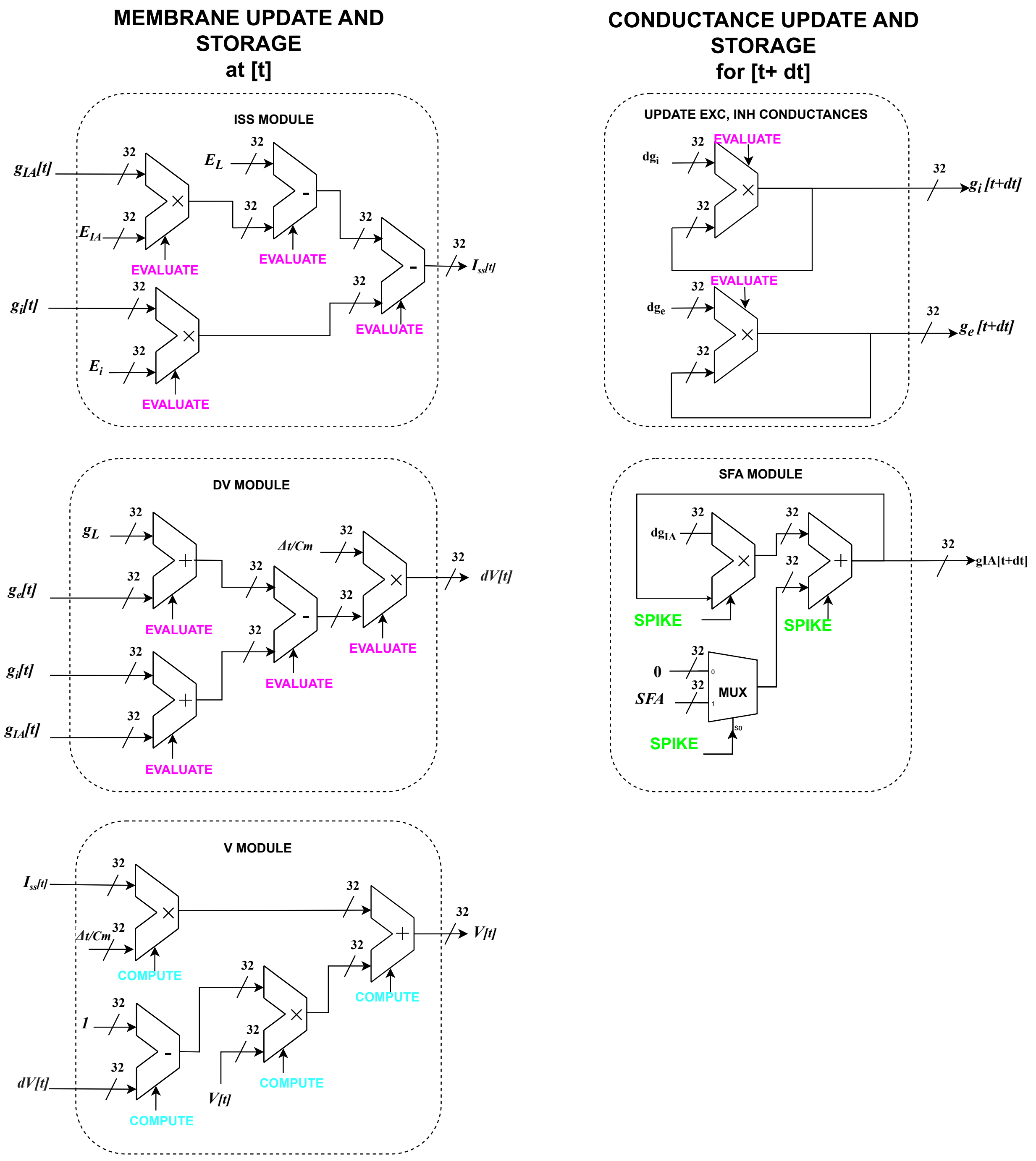


Figure 3.4: Arithmetic operations in the DGU. The left-hand side of the diagram illustrates three modules within the *Membrane Update and Storage* module: the *ISS* Module, *DV* Module, and *V* Module, responsible for computing $I_{ss}[t]$, $dv[t]$, and $V[t+dt]$, respectively. The *DV* Module and *V* Module are governed by the EVALUATE signal. The *V* Module is controlled by the COMPUTE signal. On the right-hand side, the *Conductance Update and Storage* module is depicted, responsible for computing excitatory ($g_e[t+dt]$), inhibitory ($g_i[t+dt]$), and adaptation ($g_{Ia}[t+dt]$) conductances for the subsequent timestep $[t+dt]$. This module necessitates an additional control signal (EVALUATE). Furthermore, the SPIKE signal controls the augmentation with SFA in the *SFA* module.

3.3.2. Digital control

There are two controllers, namely (i) Main Control and (ii) Refractory control. The digital control mechanism plays a crucial role in managing the floating-point operator, thereby optimizing data movements and minimizing switching activity. Notably, the inherent characteristics of neurons dictate that no spiking activity occurs during the refractory period, deactivating the arithmetic operations. Consequently, in this digital design, the respective modules and components are disabled during the refractory period, thereby minimizing switching activity. This strategy is facilitated by the controllers, which primarily activate the modules exclusively during the sub-threshold regime, effectively functioning as a computation gating technique. Figure 3.3 introduced the digital architecture of the neuron and the control signals were briefly mentioned. Before explaining the design of the controllers, we first introduce the AXI4 protocol, which is necessary to mediate the data transactions of the Xilinx arithmetic operators.

AXI4 Protocol

The AXI4 protocol orchestrates data transactions through its control signals (`tready` and `tvalid`). For input operands, `tvalid` signals initiate data transfer from the top-down, while the `tready` signals indicate the readiness of the data channel for operations. Conversely, in the output channel, these signals possess inverse definitions. In this design, the floating-point operator operates in Blocking mode, ensuring data packets are processed in pairs, akin to a queueing mechanism. Corresponding data pairs are utilized in a single operation only when all input channels have validated data available (`tvalid` for input channels and `tready` for the output channel). Thus, despite active high signals for `tvalid` in the input channels, the operation is executed and presented at the output channel only upon validation of the output channel as well. The overall delay encompasses the intrinsic latency of the IP (i.e., 9 clock cycles) and the handshake signal `m_axis_result_tready`. The diagram depicted in Figure 3.5 illustrates the protocol and the queueing effect, ensuring the correct pairing of data samples on each input channel for every operation. This mechanism is particularly beneficial in computations for membrane potential, ensuring accurate sequencing of computations.

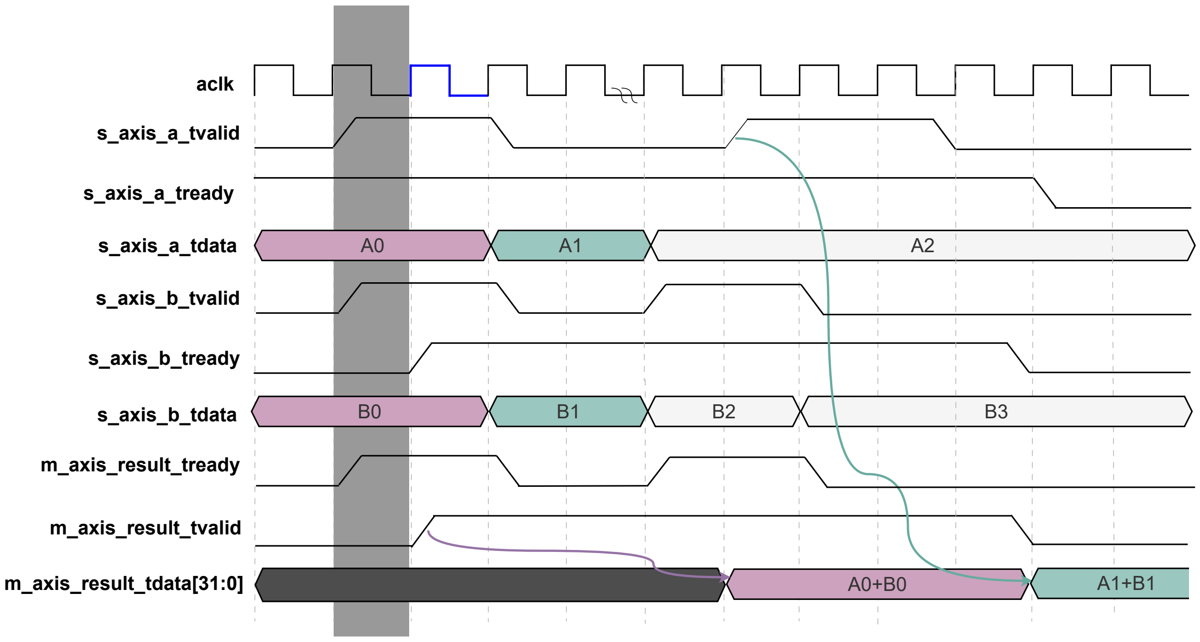


Figure 3.5: AXI4 protocol in blocking mode Starting from the left-hand side, `tready` signals from both input channels A and B are high, indicating readiness. However, the output channel remains unprepared indicated by a grey zone. Data samples are concurrently supplied through the input ports: A0, A1, and A2 for Operand A, and B0, B1, B2, and B3 for Operand B. This particular configuration is set in single-precision floating-point format and executes the addition of Operand A and Operand B. The operation and the corresponding result become available only when all control signals are active high. This synchronization occurs in the third clock cycle (blue-colored clock cycle), wherein the summation of A0 and B0 is presented following a brief delay. Subsequently, as `m_axis_result_tready` transitions to a low state in the following clock period, the output channel retains the outcome until all control signals revert to a high state. This synchronization event takes place on the sixth clock cycle. Following a similar delay, the result for the summation of A1 and B1 is made available.

Controllers

The above-mentioned control signals are generated by controllers and counters. Prior to explaining the controllers, the design of the counters are described. Due to inherent latency of the floating-point operator, there is a fixed latency, which is accounted for by utilizing counters. There are two types of counters in the design:

- **Main counter**: This counter is automatically engaged when the global reset signal is low and generates the `MAIN_COUNT` signal. Its counting operation only commences when the neuron is not in its refractory period. The main counter functions for 10,000 clock cycles (adhering to the chosen timestep of 0.1 ms), guaranteeing that all arithmetic operations are carried out and values are updated.
- **Down counter**: This counter is initiated when the neuron enters its refractory period. It proceeds to count down from the specified refractory period, which is determined by a local parameter. Once the countdown is complete, the counter is automatically deactivated.

These counters run on a clock frequency of 100 MHz, which are governed by two controllers (i) Main Controller and (ii) Refractory controller. The controllers generate a set of control signals. Their specific functions are outlined below:

- `MODE 1,0`: At the start of each timestep, provided the neuron is not in its refractory period, the *Main Control* module generates an active high for the `MODE 1,0` signal. This signal serves as a selector for the MUX, to consequently select between synaptic weights.
- `REFRACTORY`: The *Refractory Controller* issues this signal, which is exclusively set to an active high when the refractory period is active. This refractory control mechanism primarily regulates arithmetic operations, activating the FP blocks exclusively during the sub-threshold regime.
- `EVALUATE`: This control signal is asserted when the main counter reaches 13 clock cycles. The `EVALUATE` signal remains asserted during the computations of the conductances, the I_{ss} and dt.
- `SPIKE`: Although the spike signal indicates whether the neuron has spiked, it also serves as a control signal. Upon detection of a spike, a down counter is triggered in the next timestep, commencing a countdown that denotes the duration of the neuron's refractory period. It is noteworthy that in this scenario, the `REFRACTORY` signal is high, indicating active reset.
- `VALID`: At each timestep, the synaptic weight performs addition of $w_{e,i}$ and $g_{e,i}$, which is triggered by a high `VALID` signal from the *Main Controller*. Depending on the current mode (`MODE 0` for no spike and `MODE 1` for a pre-synaptic spike), the correct value for operand $w_{e,i}$ is selected.
- `COMPUTE`: This signal triggers the computation of the membrane potential $V[t + dt]$. The *Main Controller* moderates the signaling, whereby it only signals an active high after `EVALUATE` is deasserted.
- `ASSESS`: This signal triggers the assessment of the updated membrane potential $V[t + dt]$ with its resting potential V_r. This lasts while the counter is active, while all other modules are deactivated.

Figure 3.6 shows the state machine diagram of the Refractory Controller. It facilitates the control of regimes, i.e. placing the neuron in the refractory state or its sub-threshold regime via a single control signal. Furthermore, it activates the downcounter when it is in a refractory state.

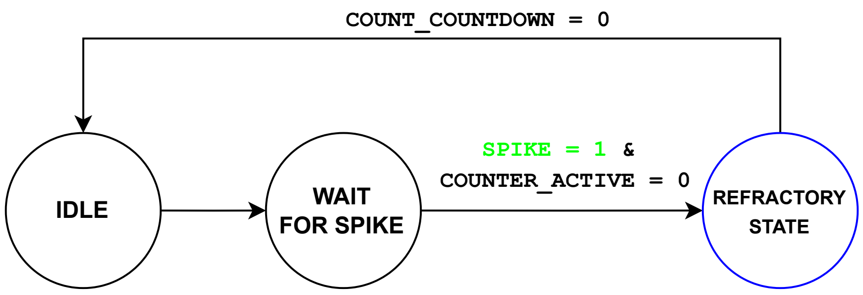


Figure 3.6: State diagram of Refractory Controller. These control signals support ensuring that the Refractory Control (de)activates the refractory state for the neuron. Upon receiving a `SPIKE`, the downcounter only initiates when the `MAIN_COUNT` is deasserted. Only, then, the countdown mechanism is activated. During the countdown, the `REFRACTORY` is high until it is done. When it is finished, it will automatically return to its Idle state. The reset signal is omitted for clarity.

Figure 3.7 shows the state machine diagram for the Main Controller. This counter initiates its operation when the neuron is not in its refractory period.

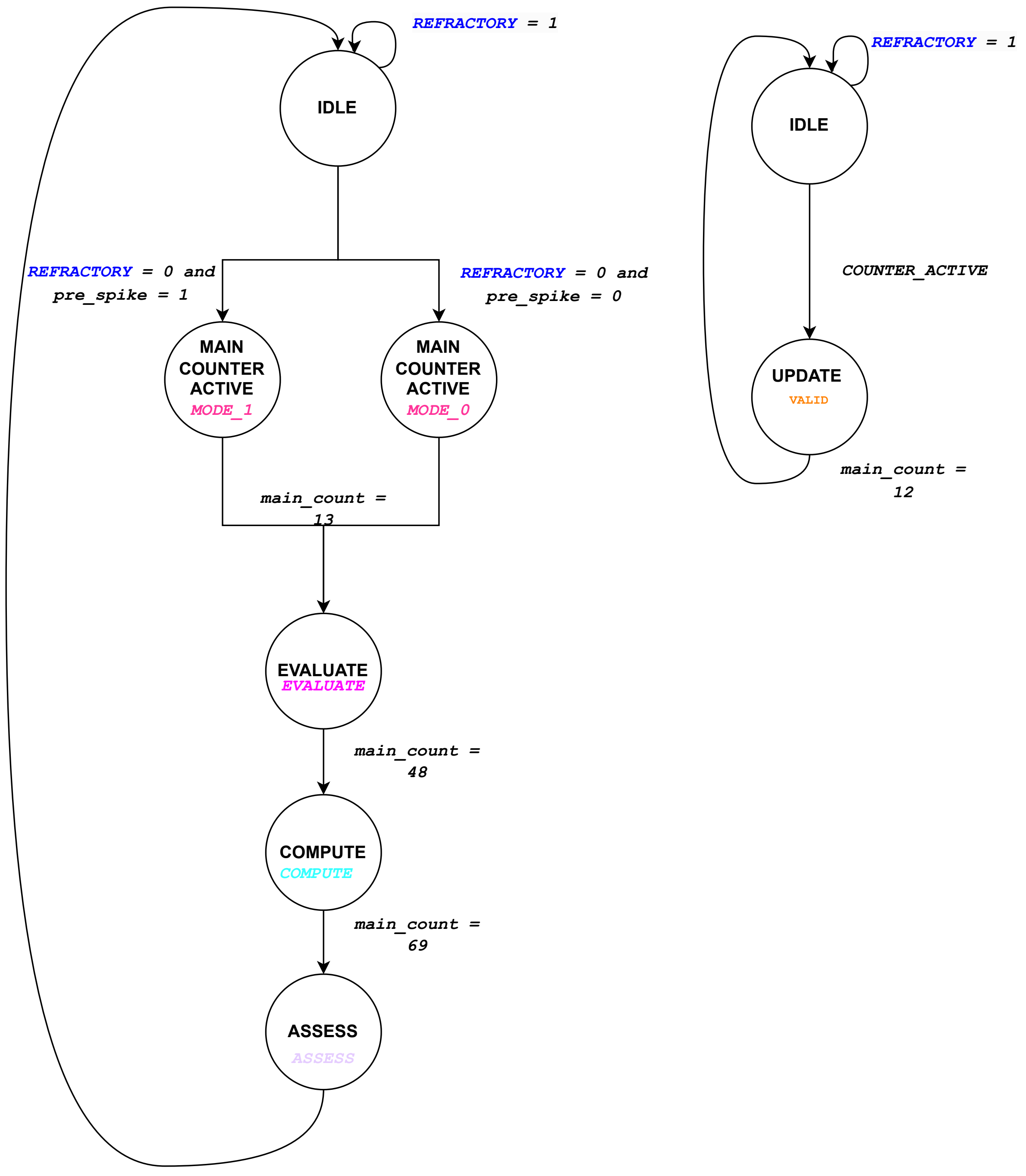


Figure 3.7: State diagram of Main Controller. When the neuron is not in its refractory state (i.e., `REFRACTORY` signal is low), the main counter is activated. The Main Controller controls all the control signals of the DGU. For the sake of clarity, the FSM on the right-hand side illustrates the state during which `VALID` signal is asserted. It is part of the Main Controller.

4 Results

This chapter presents the outcomes of the proposed software and hardware designs. Preceding the presentation of these outcomes, a comprehensive overview of the testing and verification methodology is provided. The subsequent section presents the results from the software simulations, wherein the dynamics of neuronal behavior are scrutinized at the neuron and network levels. Furthermore, this section analyzes the distinct computational primitives, clarifying their potential contributions to artificial processing within a small-scale SNN framework. Lastly, the findings from the hardware design are discussed, focusing on the overall setup and comparison of hardware and software results at a neuron level. Moreover, a conclusion is drawn on the resource utilization metrics of the proposed COBALIF neuron on a Zynq 7000 SoC FPGA platform, presenting results from a first hardware prototype.

4.1. Test and validation methodology

The testing and validation methodology comprises two distinct levels of evaluation, i.e. a single neuron and a neural network. The former is dedicated to evaluating the dynamics of individual neurons and comparing them with empirical data, aiming to assess their fidelity to real neural activity within the olfactory pathway model of the *Drosophila* larva. In contrast, the latter involves the comprehensive evaluation of the entire connectome. Both levels of methodology serve as essential baselines for conducting software and hardware tests, to validate the proposed model's efficacy and accuracy across different scales of neural activity representation. Figure 4.1 illustrates the simulation setups.

4.1.1. Neuron validation

The testing and analysis at the single neuron level entail the examination of two key parameters: (1) spontaneous spike rate, and (2) updating potential. The first parameter is instrumental in assessing the neuron's spiking activity during baseline conditions, which has been done in other works [32, 25, 29]. To validate the correspondence of the proposed SNN model to its biological counterparts, the spiking activity of individual neurons during baseline conditions must fall within predefined intervals. While the SNN model (Figure 2.8) encompasses various neuron types - including ORNs, PNs, LN, APL, KCs, MBINs, MBONs, and DANs - only the ORNs and PNs are considered for testing purposes at the neuron level. This selection is based on the fact that only for these neuron types their spiking activity have been reported as explained in Section 2.3.2.

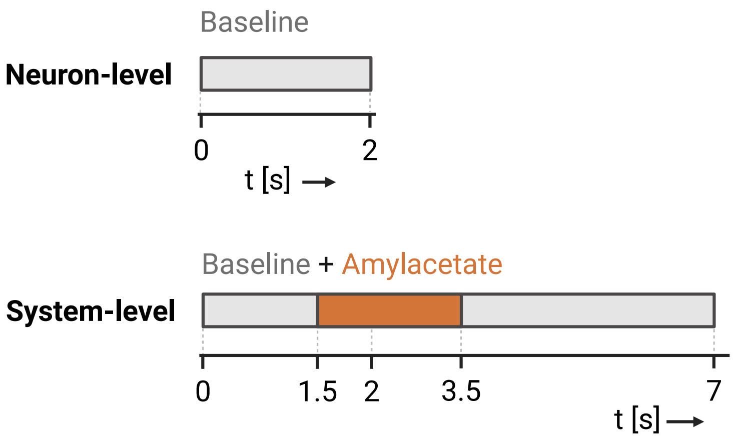


Figure 4.1: Software simulation setups. This figure illustrates the software setups for neuron and system-level validation.

The assessment of baseline activity follows the simulation protocol outlined in Jürgensen et al. [29]. Each input provided to the ORN ensemble is generated using stochastic point processes. Consequently, each ORN is linked to an ”invisible” receptor input via a one-to-one synapse and is modeled using a gamma process. The Gamma distribution, characterized by the rate and shape factor, features a baseline rate set to 65 Hz. This differs from the baseline value used in Jürgensen et al. [29]. The reason for the adjustment was made was to ensure that the baseline activity of the ORNs would satisfy the range, and to account for updated Python packages. Notably, unlike the Poisson process where the rate of event occurrence remains constant over time, the intensity function in a Gamma process may vary over time, enabling the modeling of phenomena with varying event rates more accurately. The gamma process is generated utilizing the *StationaryGammaProcess* module within the Elephant Python toolkit designed for electrophysiological software simulations [130]. The shape parameter of the gamma distribution determines the number of events being modeled, empirically set to 3.0 to approximate real-world data [45]. Additionally, the ORN response rates are approximately 5 Hz, corresponding to the average firing rate (0.2 to 7.9 Hz) of a single neuron within the ORN layer [131]. This definition is employed within the framework to compute the spiking rate, thereby necessitating an evaluation duration of 1000 ms. Point processes are utilized to represent a series of data points, often referred to as events, occurring over a defined time period. *StationaryGammaProcess* module generates these point processes, which represent the spiking inputs to the ORNs.

4.1.2. Network validation

The tests at the network level are focused on ensuring that the overall activity of the SNN adheres to the computation primitives identified in Section 2.3, i.e. spatial and temporal sparseness (or inhibition) and spike-frequency adaptation. The methodology closely follows the one proposed by Jürgensen et al. in [29]. Their procedure focuses primarily on the network level, whereby two natural odors (Amylacetate and 3-octanol) and three artificial odors are used in an associative learning experiment. The framework for the proposed solution of the thesis focuses only on the Amylacetate and compares the network response with the one from Jürgensen et al. in [29]. Amylacetate has been used in various odorant emulations and simulations as it is one of the fundamental odors

to *Drosophila* larva [132]. The odor is generated with the StationaryGammaProcess and each of the 21 ORNs receives a unique spiking pattern that simulates the spiking activity of the sensory inputs.
The initialization and parameters of the neurons can be found in Appendix B.

4.2. Software validation

This section focuses on the software results following the two-stage process as explained in the previous section. Firstly, the neuron validation is presented, whereby the spiking activity of the ORN and PN are shown. Secondly, the software model is validated on the network level.

4.2.1. Neuron validation

For the single neuron validation, an input spike train is generated between zero and 1000 ms (10,000 timesteps), as explained above. We will have a look at the temporal evolution of the membrane potential as well as the excitatory and adaptation conductances. Firstly, in Figure 4.2, the baseline activity of the ORN and PN are shown for Brian 2 and Lava simulation runs. This figure shows that the Lava results correspond with those of Brian 2, concluding that the algorithm to update the membrane potential (see Algorithm 1) is successfully implemented in the Lava framework. Furthermore, we can tell from this figure when the ORN eventually spikes. This is the case when the membrane potential rapidly decreases to its resting potential ($V_r = -60mV$). Therefore, the ORN generated 5 spikes within the simulation time of 1000 ms, which satisfies the baseline firing frequency. Moreover, from the plots of the conductances of the ORN and PN, we can conclude that the two simulation models match and are within the expected range, i.e., 0.2 to 7.9 Hz [45].

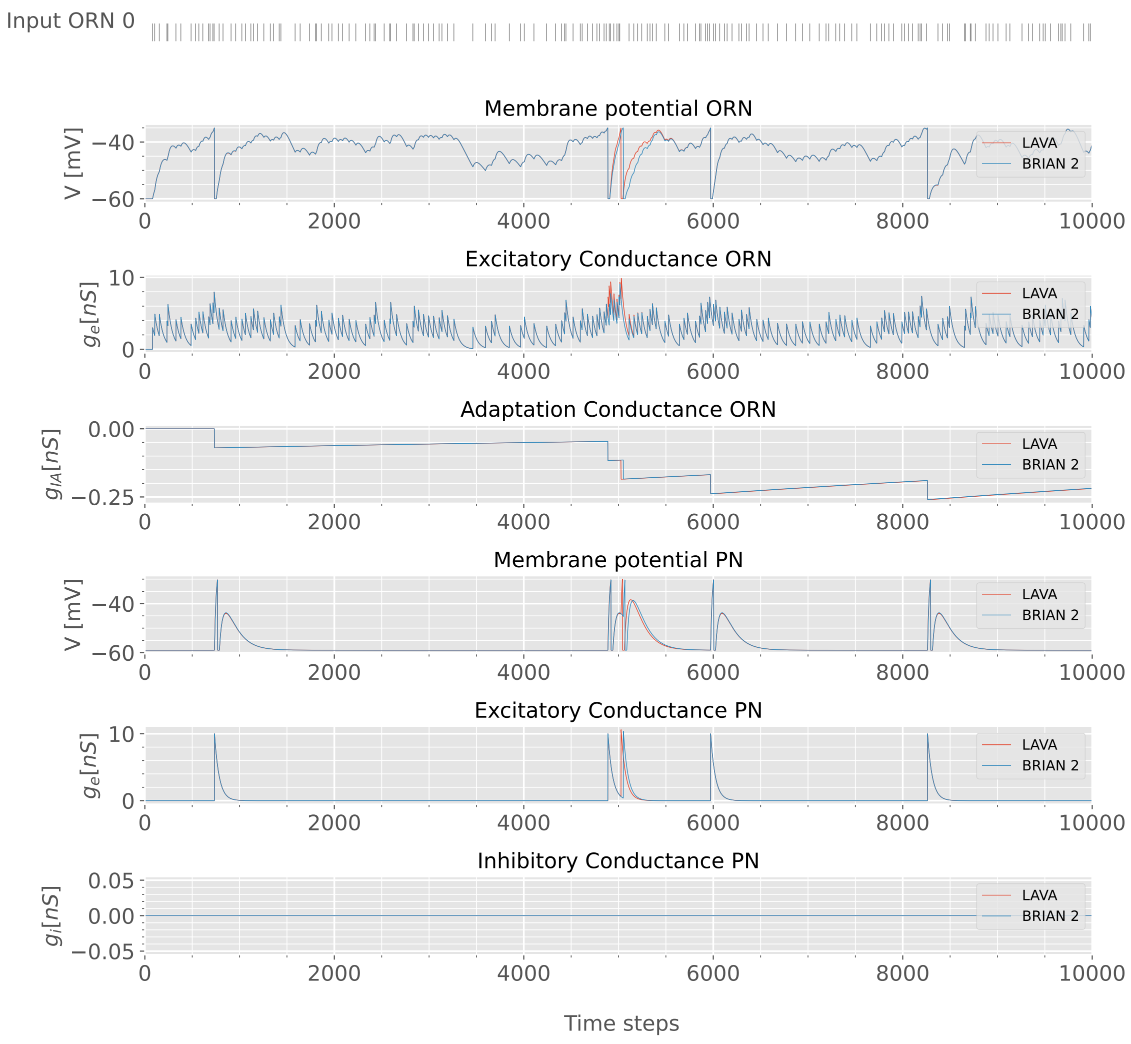


Figure 4.2: Response of the ORN and PN in baseline activity. The ORNs and PNs are subjected to a baseline activity, whereby spikes are generated with the *StationaryGammaProcess* function. The input is related to the first ORN type (i.e, ORN 0) and thus we only consider this ORN and its cognate PN within the respective glomerulus. The spiking activity is 5 Hz.

4.2.2. Network validation

For network-level validation, we are presenting an odor input (i.e., Amylacetate) to the model, which is modeled by the StationaryGammaProcess. In this case, the model now also comprises LNs for lateral inhibition, KCs for sparse encoding, and the APL for inhibiting the spiking activity of the KCs. The odor is presented between 1500 and 3500 ms in a 7s simulation. According to the simulation procedure in Figure 4.1, we validated the network model in Brian 2 and Lava. Figure 4.3 shows that during odor presentation, the activity and network response increases. This happens in the ORN layer, which is subjected to SFA. The Lava model closely follows the Brian 2 model, which is specifically presented in the adaptation conductance curve. At the odor onset, the ORNs rapidly increase their activity, which can be derived from the potential membrane plots for both neuron types. Consequently, the PNs exhibit a strong activation during odor onset concerning its baseline activity with only 5 spikes/s. During odor presentation, we also see a strong increase in inhibitory conductance of the PN, which accounts for the lateral inhibition.

Figure 4.4 shows the spike raster plots of the ORNs, PNs, and LNs. Additionally, on the right-hand side, plots are shown of the global spike count. Both models demonstrate the effect of spike frequency adaptation - applied to the ORNs only - which results in a decay of global spike counts. At odor onset, we find the highest count of spikes, which gradually decreases. The difference between global spike counts is particularly large in the PN layer, i.e. around 60 spikes. This is because Lava processes the incoming excitatory and inhibitory signals in different timesteps, rather than simultaneously which is the case in Brian 2. This could be resolved by creating a specific `ProcessModel` that can account for these changes. ORNs and LNs only receive one type of neurotransmitter input (i.e., excitatory). In these cases, we do not find a large discrepancy.

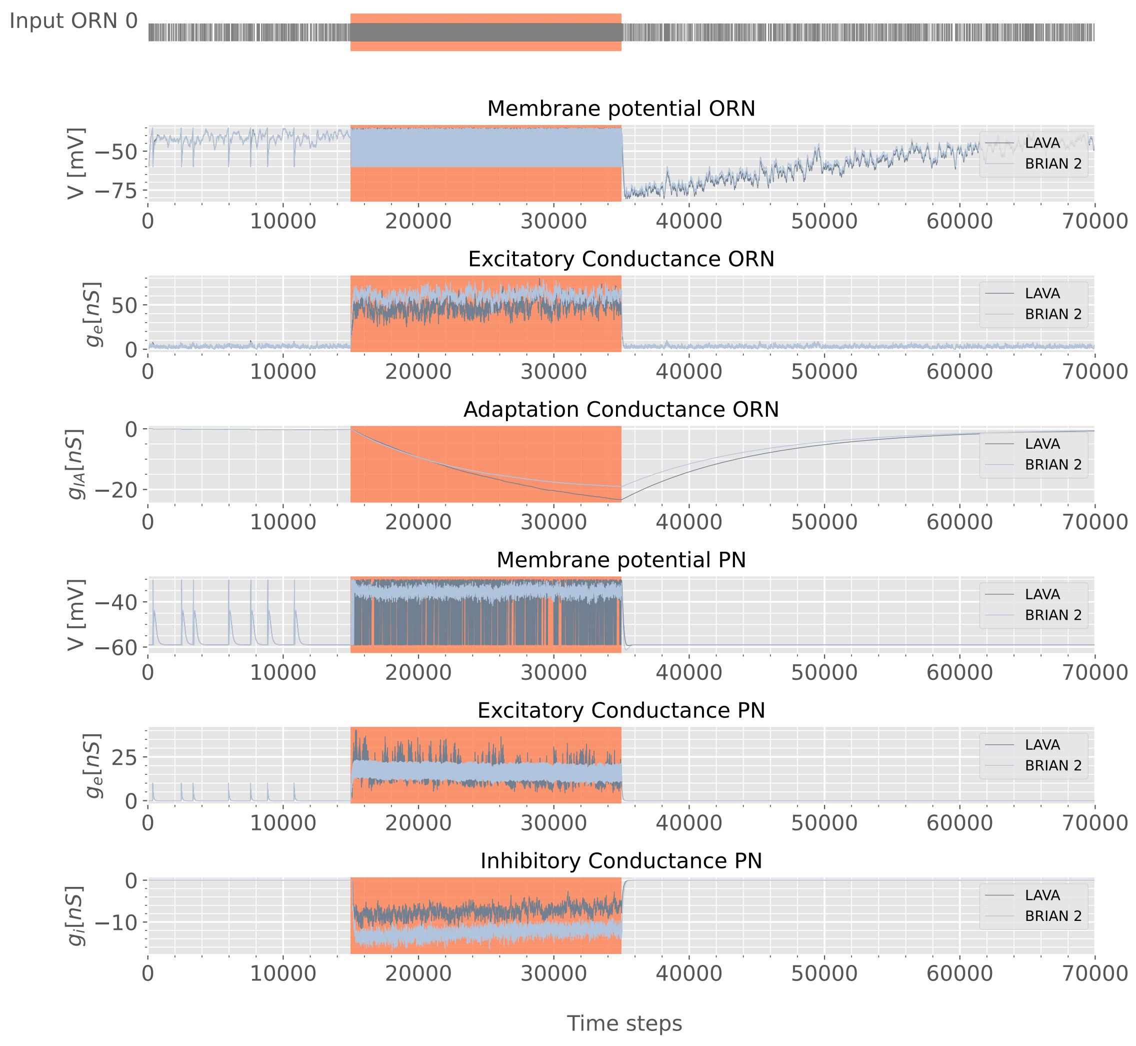


Figure 4.3: Amylacetate network responses for PNs and ORNs. The odor is encoded in orange in the figure. The figure includes the average time-varying responses of the PNs and ORNs, consisting of the membrane potential, excitatory conductances, the inhibitory conductance of the PNs, and the adaptation conductance of the ORN.

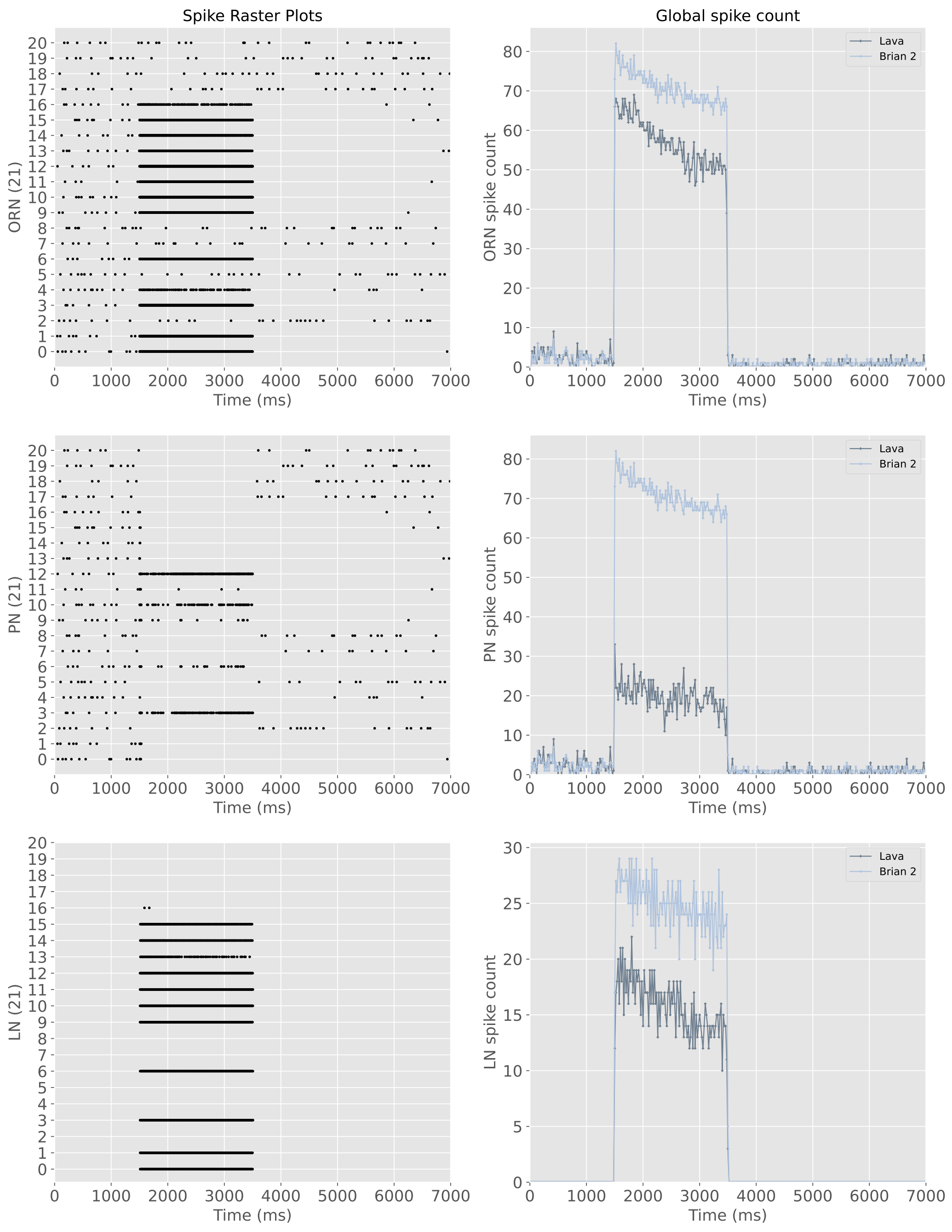


Figure 4.4: Network response of ORNs, PNs and LNs to Amylacetate for Brian 2 and Lava. The figure shows on the left-hand side the spike raster plots of the ORNs, PNs, and LNs. On the right-hand side, the global spike count of these neuron populations is shown to show their activity during odor presentation.

To fully cover the network model without the learning part, we also simulated the KCs Figure 4.5 shows the spiking raster of the KCs for both simulation models. The KCs in both simulations are heavily inhibited by the APL to ensure reduced spiking activity. The spike raster plot effectively encodes the sparse spatio-temporal code for the Amylacetate odor. The spiking activity is sparser in the Brian 2 simulation, which is essentially due to the same cause as explained previously. However, the overall level of sparseness for both models is in agreement with empirical data [29].

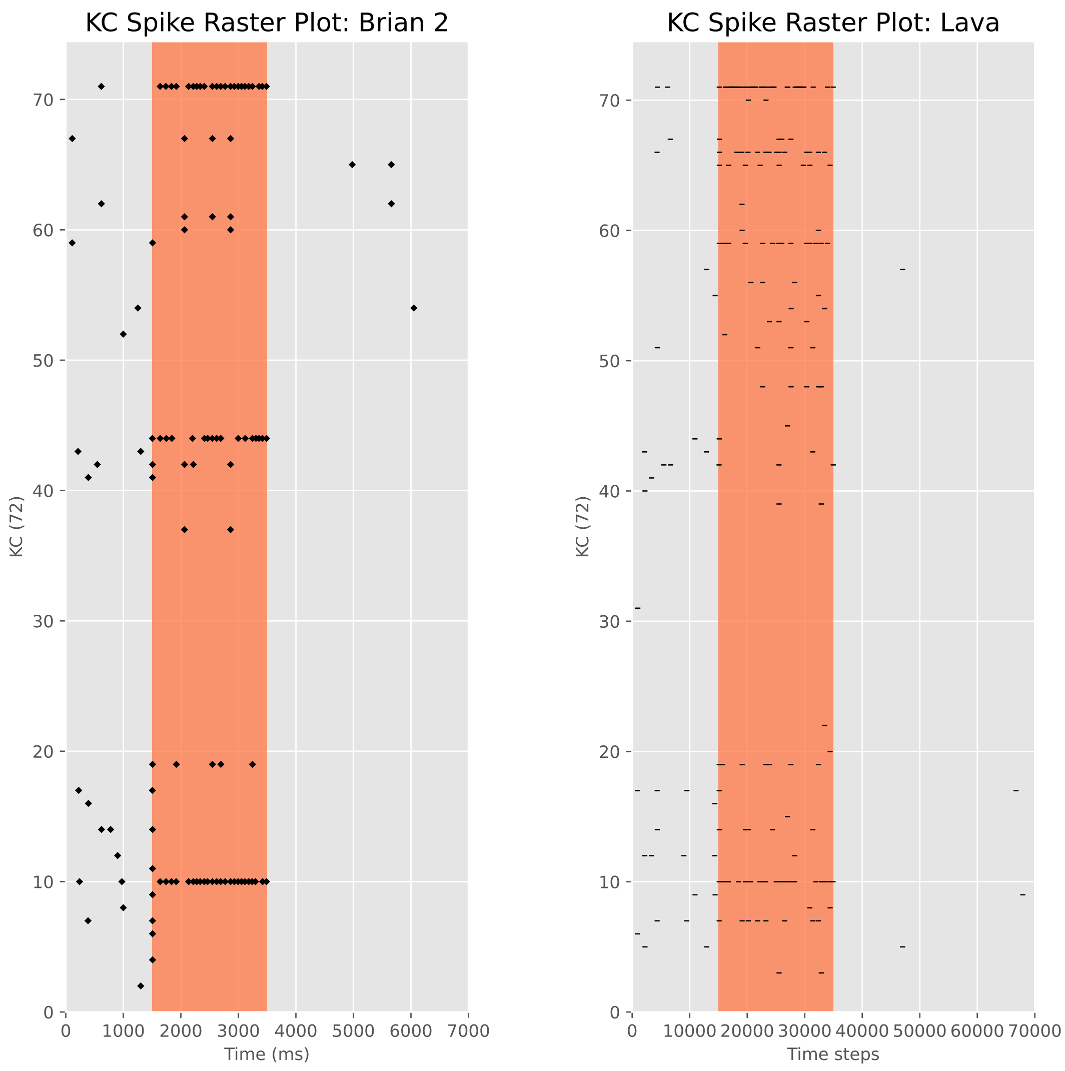


Figure 4.5: Response of Kenyon cells during odor presentation. The odor is presented in orange.

4.3. Hardware validation

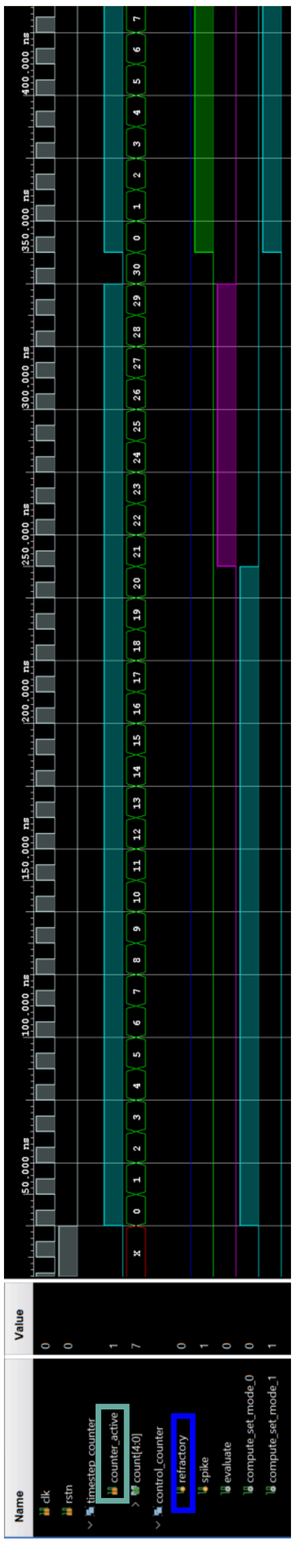
The development and validation of the hardware design concentrate on the design of the ORN. The digital design of the COBALIF neuron is introduced Chapter 3, wherein all conductances are incorporated. However, for the ORNs, only the excitatory and adaptation conductance modules are required, with the inhibitory conductance block being omitted. This section begins by presenting the validation of the controllers and counters, assessing the timing and control signals. Subsequently, we explain the neuronal dynamics, encompassing the updating of the temporal components as outlined in Algorithm 1. Lastly, an overview of the utilization of the resources are presented and analyzed.

4.3.1. Control simulations

The two controllers (i.e., Main Controller and Refractory Controller) are validated through examining their post-synthesis simulation results with the expected states and values from their respective FSMs.

Main controller

The Main Controller is the top-level controller that checks if a digital neuron - at the start of the timestep - is in refractory state or in sub-threshold regime. This is signaled by a neuron-specific refractory signal (`REFRACTORY`). In case the neuron is in sub-threshold regime, the main control activates the main counter for the respective neuron at the start of the timestep. Contrarily, in refractory state, the counter should not be activated. In Figure 4.6, two post-synthesis simulation waveforms present these two different cases. Figure 4.6 demonstrates that in sub-threshold regime, the counter is activated by triggering the `COUNTER_ACTIVE` signal. It also immediately checks if a pre-synaptic spike (`PRE_SPIKE`) occurs during the first count. This drives the signaling of the `MODE_0` and `MODE_1` signals, indicating if a spike is detected or not, respectively. The state machine for the main control asserts the `EVALUATE` signal, initiating computations for the current simulation time according to Algorithm 1 in the *Membrane Update and Storage* module .
In the second case, for a new timestep, a spike does occur during the *evaluate* state, which triggers an active high for the `REFRACTORY` signal. The main counter is then in refractory state, where the counter is quiescent. These simulations show that the FSM specifications - as described in Subsection 3.3.2 - are met.



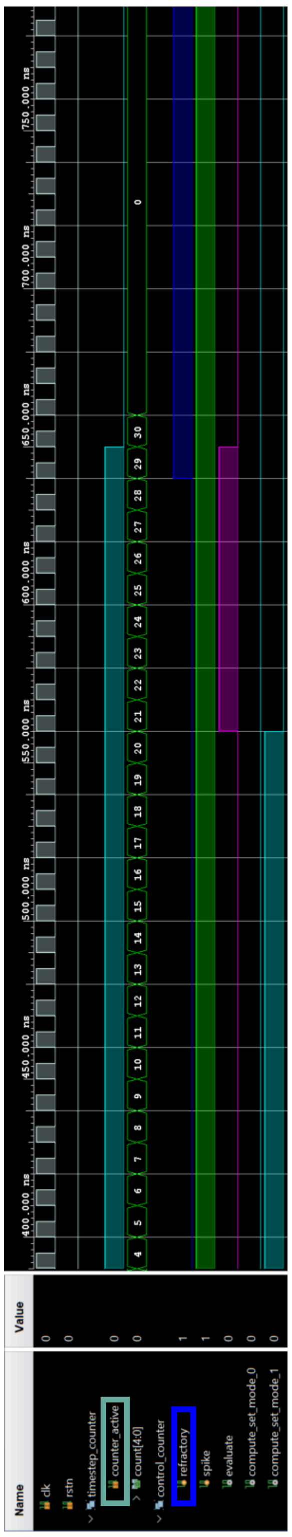


Figure 4.6: Post-synthesis simulations for Main Control. The figures represent the neuron in (i) sub-threshold regime and (ii) in refractory period. In the first case, the main counter is activated, because REFRACTORY is low, i.e., the neuron is not in refractory state. The Main Control generates an active high `counter_active` signal. Contrarily, in the second case, the `counter_active` is low as the neuron is its refractory period.

Refractory controller

The Refractory Controller is in charge of two crucial mechanisms (i) activating the count down module, and (ii) moderating the comparator. Figure 4.7 shows the validation of the count-down activation triggered by the refractory state controller. The refractory counter employs a countdown mechanism that is equivalent to 64 clock cycles. It is controlled by the refractory state machine, for which Figure 4.7 shows that it is activated when a spike occurs when the main counter is at count 29. This value is taken to be 64 for illustrative purposes, but it can be adjusted according to the chosen refractory time. Since these simulations are only focused on single-neuron design. This suffices to demonstrate the operatibility of the countdown mechanism.

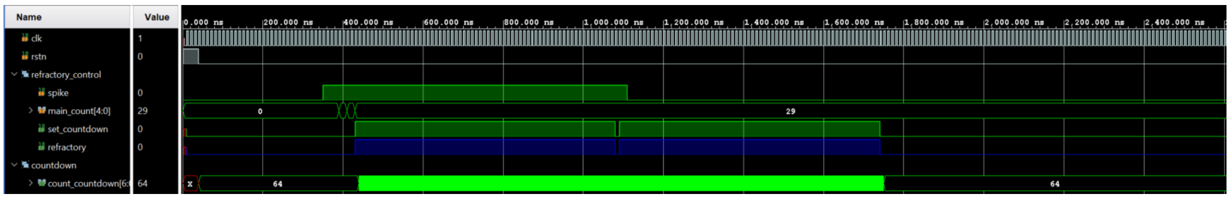


Figure 4.7: Post-synthesis simulations of the activation of the countdown module. This figure presents the activation of the count down, when a spike has been generated.

In Figure 4.7 for the main_count signal other numbers such as 28 and 30 were given, which are the two small signal changes before the yellow line on the `main_count` line. As expected and according to the FSM - it does not trigger the refractory controller. Only when the main counter is 29, the count down from 64 to zero starts.

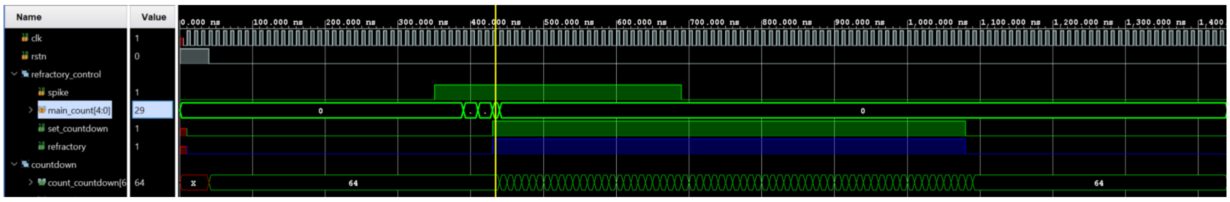


Figure 4.8: Post-synthesis simulations for refractory state control (Close-up). This is a close-up showing that the FSM of the refractory control is satisfied, whereby the state in which it is counting down cannot be obstructed.

In Figure 4.8 even when the main count goes back to zero - according to how the main counter would go back after reaching count 29 - the countdown mechanism still remains active and continuous to count down, starting from 64 to 63 onwards. Note that also at some point in time, the spike goes to zero, but still, the countdown remains counting.

The *Refractory Controller* also activates the comparator after all neuronal computations have been executed. After computing the $V[t + \Delta t]$, that value will be given as an input to the comparator. This value is compared with the resting potential of the respective neuron. When this membrane potential is equivalent to the resting potential, the comparator will generate an active high for the `REFRACTORY` signal, putting the neuron in its refractory state. Figure 4.9 demonstrates this operation.

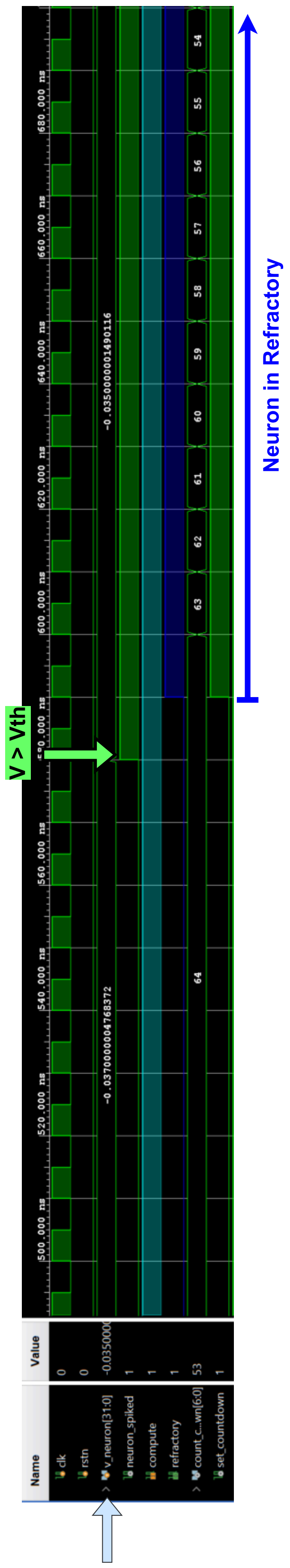


Figure 4.9: Post-synthesis simulation upon exceeding the membrane potential. The blue arrow indicates the membrane potential and the green arrow indicates the point at which the neuron potential is equivalent to the threshold voltage. Consequently, the REFRACTORY signal is triggered, putting the neuron in a refractory state.

4.3.2. Neuronal dynamics

The above-mentioned controllers moderate the execution of the arithmetic operations and their timings. Based on the digital architecture design of the neuron, the validation of the neuronal dynamics was carried out in three phases: (i) updating the pre-synaptic weight, (ii) updating the ISS and DV, and (iii) computing the membrane potential for the next timestep according to the Forward Euler technique. Each phase is controlled by control signals from the Main Controller. In the first phase, the associated control signals are `VALID`. The second and third phases are controlled by `EVALUATE` and `COMPUTE`, respectively. In order to maintain a readable resolution of the waveforms, the full-scale timestep of 0.1 ms is not shown in the following figures.

Figure 4.10 shows how the pre-synaptic weight module updates the current excitatory conductance upon receiving a spike, concerning Case (i). This is triggered by the `VALID`. Only when a pre-synaptic occurs, the excitatory conductance will be updated. The figure demonstrates the correct functionality of the `VALID`.

Figure 4.11 shows the post-synthesis simulation results, targeting the membrane potential for an ORN. This figure serves as a validation of Case (ii) and Case (iii). The ORN has two conductance elements, namely excitatory and adaptation. This figure demonstrates how the membrane potential is updated during its sub-threshold regime. Given that the neuron has not spiked yet, its adaptation weight will remain zero throughout the simulation. The simulation is divided into five periods (T1, T2, T3, T4, T5). The neuron receives an excitatory, pre-synaptic spike in the odd-numbered periods. According to the FSM, it will first update the excitatory weight by adding the synaptic constant upon spike. This is triggered by the `VALID` signal. Then, the main control will give an active high for `EVALUATE` signal, initiating the computations of the components ISS and DV in the *Voltage Update and Storage Module*. Then, the signal `COMPUTE` is activated, which allows for the computation of the membrane potential in the next timestep, i.e. $V[t + \Delta t]$. The blue arrows in the figure indicate that the updated membrane potential is computed for and available in the next timestep.

Regarding the validation of the computed values, we present a table of the expected values for each parameter in the algorithm and the generated values using the digital neuron design. In the table Table 4.1, we can verify that the overall arithmetic operators perform according to the state machines and perform the correct operations in the correct order. This ensures the correct operability of the neuron in the sub-threshold regime.

Table 4.1: Comparison of software and hardware (in purple) computations results in sub-threshold regime for ORN 0. A spike is indicated with a green colour.

Timestep	ge[t] [nS]		Iss[t] [A]		dv[t][-]		V[t] [mV]		V[t+dt] [mV]	
🟩 **T1**	3.0000	3.0000			0.00800	0.007999	-0.060000	-0.0599999	-0.059820	-0.059820
T2	2.9400	2.9400			0.007940	0.007939	-0.0598200	-0.0598200	-0.059645	-0.059645
🟩 **T3**	5.8812	5.8812	3×10^{-10}	3×10^{-10}	0.0108112	0.01081199	-0.05964500	-0.05964503	-0.059296	-0.059296
T4	5.7635	5.7635			0.0107600	0.0107635	-0.05929599	-0.05929602	-0.058957	-0.058957
🟩 **T5**	8.64830	8.648305			0.0136483	0.0136483	-0.0589577	-0.0589577	-0.058453	-0.058453

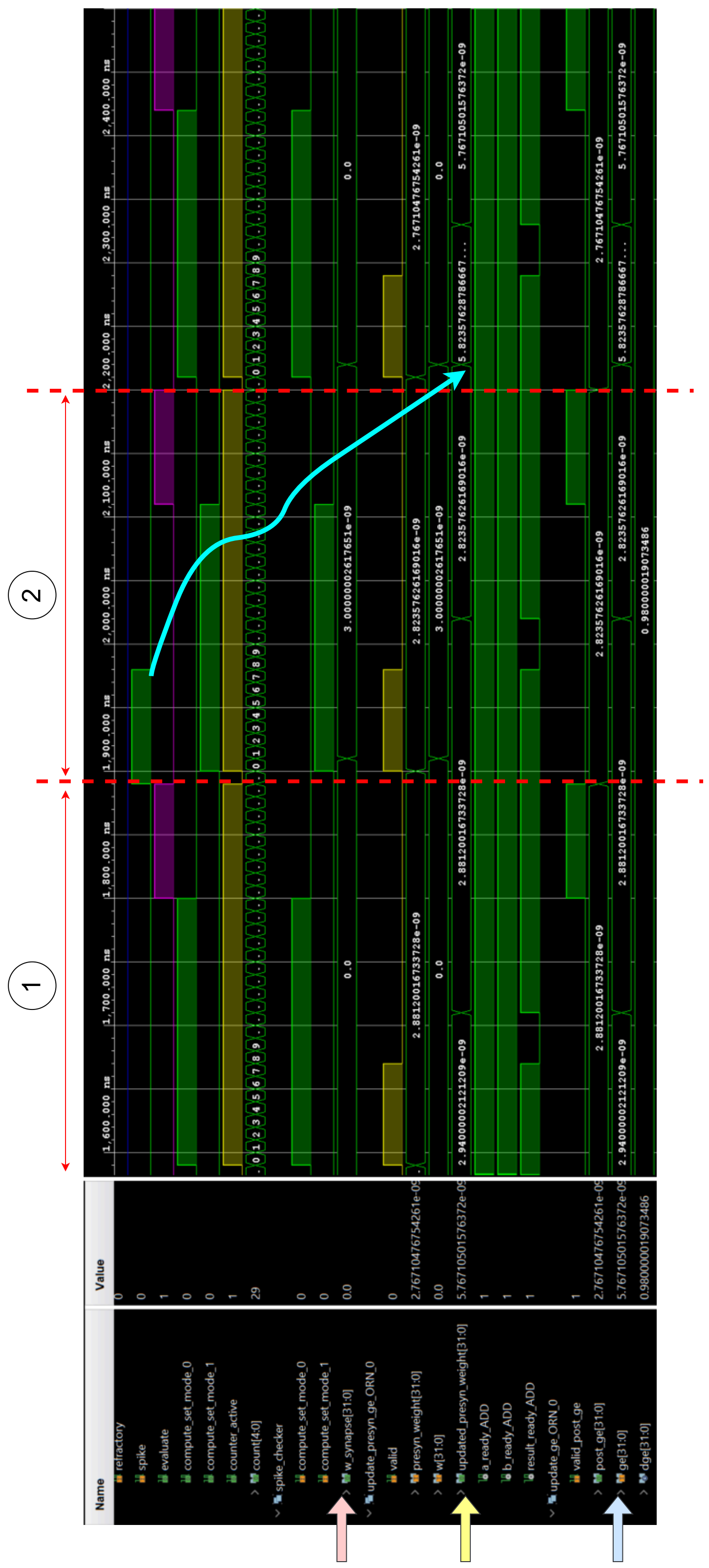


Figure 4.10: Case i: Updating the excitatory conductance of a post-synaptic neuron due to a pre-synaptic spike. This simulation demonstrates the working principle of updating the excitatory conductance (or excitatory synaptic weight) for two cases: (1) no spike, and (2) a pre-synaptic spike occurred. The pink arrow shows that in Case (1) the excitatory conductance performs a summation with 0, but in Case (2) it selects the weight to update the excitatory conductance strength. The blue arrow - pointing at `updated_presyn_weight` - shows that this update weight passes through the multiplication before it updates the `updated_presyn_weight`. This value is instantly available in the next timestep. Hence, the yellow arrow shows that this selected weight ($3\ nS$) is added to the current excitatory conductance of the respective post-synaptic neuron. The blue arrow crossing the figure indicates that the final updated value is visible at the start of the next time step.

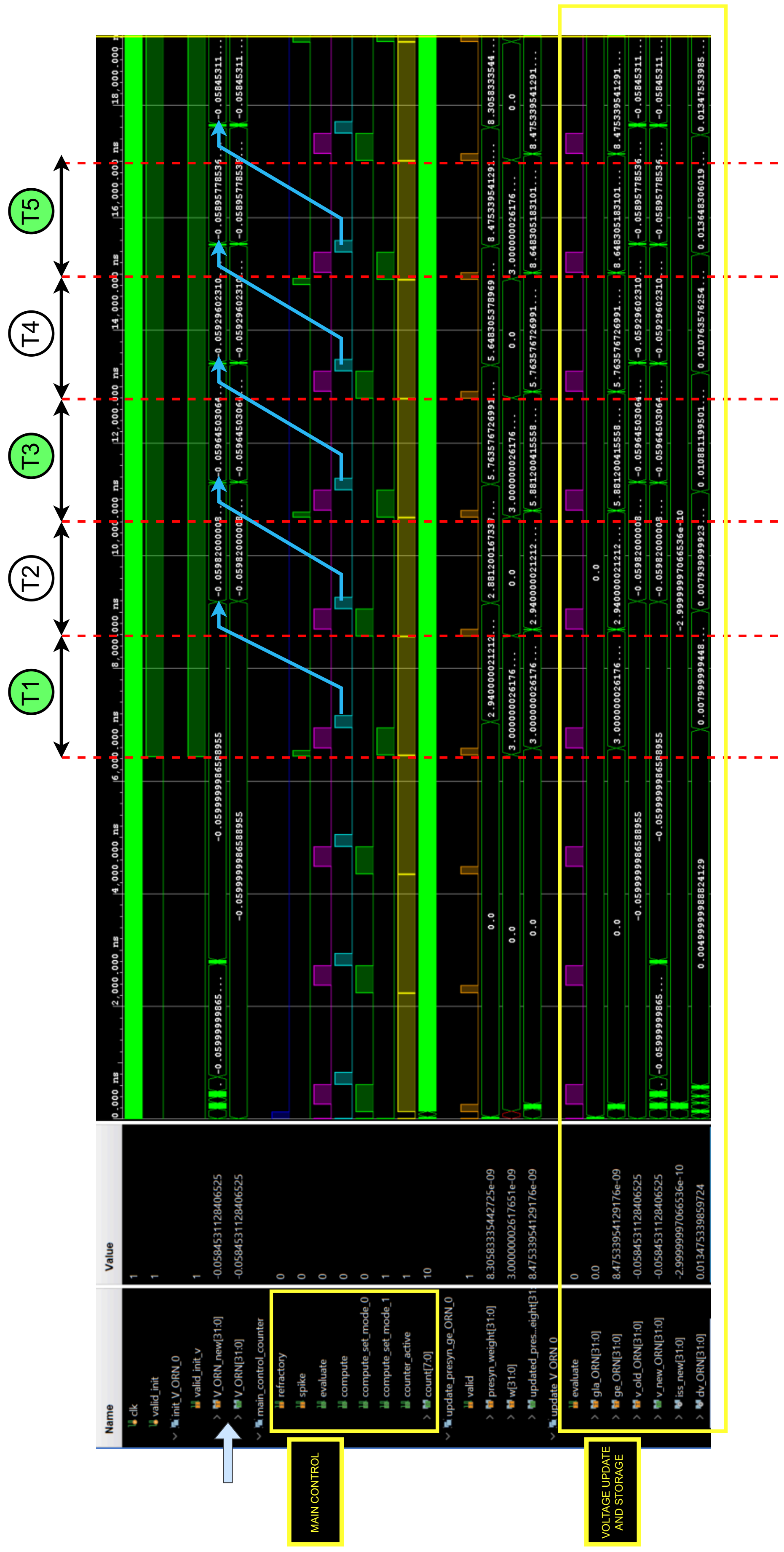


Figure 4.11: Case ii and iii: Computing ISS and DV, and $V[t+\Delta t]$ using the ***Voltage Update and Storage Module***.

As the timestep for updating membrane voltage is 0.1 ms (Δt), Figure 4.12 serves as demonstration of how the membrane potential updates every timestep, using the exact same mechanism as explained above. This figure confirms that the neuronal state is computed on a time-step basis, which is similar to the software design.

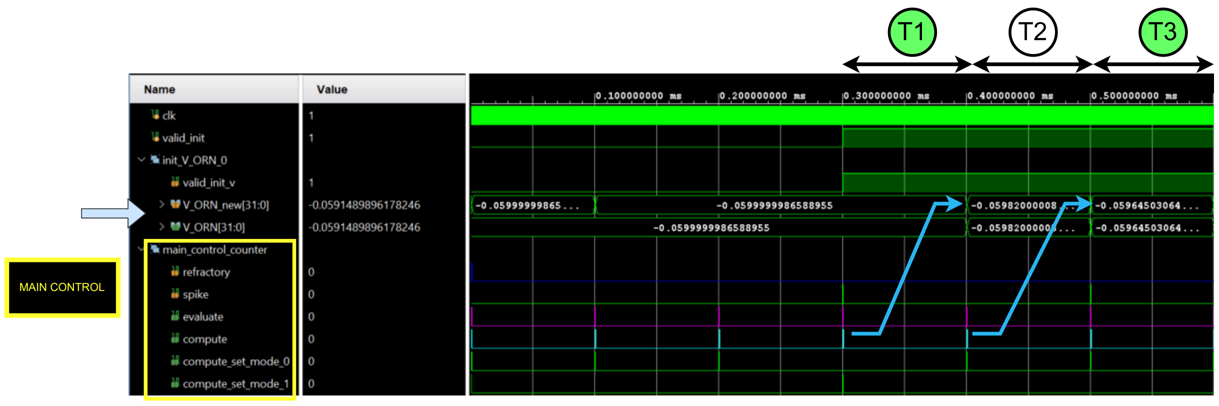


Figure 4.12: Updating $V[t+\Delta t]$ in timesteps of 0.1 ms using the *Voltage Update and Storage Module*.

4.3.3. Resource utilization

For a single ORN, the post-implementation resource utilization is summarized in Table 4.2. This table indicates the allocations of the look-up tables (LUTs) and flip-flops (FFs). Additionally, there are the IO (related to the input and output ports of the FPGA) and the BUFG (global clock buffer). Table 4.2 shows that for a single neuron (utilizing single-precision floating-point operators) the resource utilization is low, i.e. 0.13% for LUTs and 0.13% for FFs. Consequently, it is possible to scale the design to about 750 COBALIF neurons. As for the IO and BUFG, these are more global, architecture-independent parameters. The former accounts for the four input signals (including the system clock, system reset, an artificial spike input, and an additional initialization signal) given during simulations, which are automatically interpreted as real input ports during synthesis and implementations. To account for this, the on-board BRAM can be used to store input signals and specifically input spikes such as the odor-specific spike trains. The latter was instantiated by the Vivado synthesis and implementation tools, and may not scale with the number of neurons as it accounts for a global, reusable signal.

Table 4.2: Post-implementation resource utilization of a single digital COBALIF neuron

Resource	Utilization	Available
LUT	67 (0.13%)	53200
FF	143 (0.13%)	106400
IO	4 (3.20 %)	125
BUFG	1 (3.13%)	32

5
Conclusion and Future Work

Numerous studies have been dedicated to small spiking neural networks designs employing efficient neural processing with a minimal power consumption. The olfactory pathway model of the *Drosophila* is a well-studied example of a low-footprint system, capable of performing chemo-sensory processing including learning, and anticipating their behaviour accordingly *on the fly*. Consequently, many studies have conducted research to explore its neural architecture and processing. This thesis proposes a phenomenological conductance-based leaky integrate-and-fire (COBALIF) neuron, which is used in a digital solver design on a Zynq 7000 SoC. It is followed by synthesis and implementation, accounting for approximately 67 LUTs and 143 FFs for a single neuron, showing that scaling to $\sim$750 neurons is possible with the selected FPGA. We summarize below the key takeaways of this work addressing the main research question:

1. The proposed computational model of the COBALIF neuron provides a feasible solution for emulating and preserving the spatio-temporal dependencies of the neural behaviour in the olfactory pathway model of the *Drosophila* larva.
2. A digital circuitry comprised of arithmetic operators that solve partial differential equations using single-precision floating-point arithmetic can yield results that align with those obtained through software, while maintaining accuracy across a dynamic numerical range.
3. Utilizing digital controllers as data gating mechanisms on the selected FPGA can serve to mitigate high data traffic and transitions in time-multiplexed designs, without sacrificing the granularity level.

The proposed model for the COBALIF neuron has demonstrated successful computational processes, updating the neuron's state at intervals of 0.1 ms. This model lays a foundation for scaling up to small-scale SNN networks. The model was firstly benchmarked on Intel's Lava framework using Python to validate the design at spiking neuron- and network-level. To this end, the three stages of the olfactory pathway model of the *Drosophila* larva were modeled. Through verification with experimental data, including baseline activity and odor presentation with Amylacetate, we employed an algorithm utilizing Forward Euler approximation without tuned temporal parameters. Comparison of spiking activities between Lava and its prior-version in Brian 2 highlighted similarities. To evaluate the feasibility of maintaining the same level of accuracy of the spatio-temporal concepts of the COBALIF neuron, a digital counterpart of the neuron model was developed. This digital neuron unit integrates digital solvers for differential equations and controllers to minimize data movement during refractory states. We validated the digital COBALIF neuron design through post-synthesis simulations in Vivado at a clock frequency of 100 MHz. The computations were compared with software results, confirming maintained accuracy compared to software results. The event-driven updating of the neuron ensures fine granularity. Further validation of the FPGA-based digital design is necessary to assess its potential deployment on FPGA during the post-implementation phase. This continued validation from a hardware perspective would contribute significantly to our understanding of its practical application and scalability.

To further elaborate on the scope and significance of these takeaways, our research explores potential avenues and areas for future work:

- **Full-scale modelling on the FPGA**: In future work, given that we can scale to approximately 750 COBALIF neurons, we can implement the entire olfactory pathway model. Binary odor inputs can be stored in the BRAM of the FPGA. The validations of the hardware design can then be validated on the FPGA board, whereby the FPGA acts as a stand-alone device that is able to update the state of the neuron in the population for any given input.
- **Neuromodulation mediating online learning**: The learning part of the olfactory pathway model has been discarded in this thesis. In future work, the learning through neuromodulatory neurons can be simulated. Due to the modularity of this design, the type of neurons can be expanded to accommodate for dopaminergic neurons and employing their learning rules on the FPGA board. To address bio-plausible operations at the neuron level within a footprint suitable for edge computing applications, the digital neuron design could be also used for solely dynamic learning solutions. While this thesis primarily focused on prototyping olfactory-inspired neurons, future efforts could delve into exploring learning principles, such as prediction error coding inspired by neuromodulation in the mushroom body.
- **COBALIF neuron on neuromorphic hardware**: As the COBALIF neuron has been both validated in hardware and software (i.e., Lava framework), its computational model can be programmed in microcode to benchmark it on neuromorphic hardware, i.e., Loihi 2. This would be a potential avenue to implement the model in a fully neuromorphic platform, which would circumvent the usage of any time-multiplexed processes, offering a fully event-based implementation.

In summary, this thesis offers an initial, compelling prototype in olfactory pathway modeling. The proposed neuron model utilizing digital circuitry may serve as a building block for expansion and the outline of future work provides a framework to continue the exploration of this design on small network level.

References

[1] Malcolm Burrows. “The neurobiology of an insect brain”. In: (1996).

[2] Randolf Menzel et al. “Cognitive architecture of a mini-brain: the honeybee”. In: *Trends in cognitive sciences* 5.2 (2001), pp. 62–71.

[3] Philipp D Lösel et al. “Natural variability in bee brain size and symmetry revealed by micro-CT imaging and deep learning”. In: *PLoS Computational Biology* 19.10 (2023), e1011529.

[4] Carver Mead. “Neuromorphic electronic systems”. In: *Proceedings of the IEEE* 78.10 (1990), pp. 1629–1636.

[5] Carver Mead. “How we created neuromorphic engineering”. In: *Nature Electronics* 3.7 (2020), pp. 434–435.

[6] Catherine D Schuman et al. “Opportunities for neuromorphic computing algorithms and applications”. In: *Nature Computational Science* 2.1 (2022), pp. 10–19.

[7] Thomas Dalgaty et al. “Insect-inspired neuromorphic computing”. In: *Current opinion in insect science* 30 (2018), pp. 59–66.

[8] Thomas N Theis et al. “The end of moore’s law: A new beginning for information technology”. In: *Computing in science & engineering* 19.2 (2017), pp. 41–50.

[9] Hadi Esmaeilzadeh et al. “Dark silicon and the end of multicore scaling”. In: *Proceedings of the 38th annual international symposium on Computer architecture*. 2011, pp. 365–376.

[10] Charlotte Frenkel et al. “Bottom-Up and Top-Down Approaches for the Design of Neuromorphic Processing Systems: Tradeoffs and Synergies Between Natural and Artificial Intelligence”. In: *Proceedings of the IEEE* (2023).

[11] Olli J Loukola et al. “Bumblebees show cognitive flexibility by improving on an observed complex behavior”. In: *Science* 355.6327 (2017), pp. 833–836.

[12] Alex C Keene et al. “Drosophila olfactory memory: single genes to complex neural circuits”. In: *Nature Reviews Neuroscience* 8.5 (2007), pp. 341–354.

[13] Peter Mombaerts et al. “Visualizing an olfactory sensory map”. In: *Cell* 87.4 (1996), pp. 675–686.

[14] Michael Winding et al. “The connectome of an insect brain”. In: *Science* 379.6636 (2023), eadd9330.

[15] Matthew E Berck et al. “The wiring diagram of a glomerular olfactory system”. In: *Elife* 5 (2016), e14859.

[16] Timo Saumweber et al. “Functional architecture of reward learning in mushroom body extrinsic neurons of larval Drosophila”. In: *Nature communications* 9.1 (2018), p. 1104.

[17] Bertram Gerber et al. “The Drosophila larva as a model for studying chemosensation and chemosensory learning: a review”. In: *Chemical senses* 32.1 (2007), pp. 65–89.

[18] Heike Demmer et al. “Intrinsic membrane properties and inhibitory synaptic input of kenyon cells as mechanisms for sparse coding?” In: *Journal of Neurophysiology* 102 (3 Sept. 2009), pp. 1538–1550. DOI: 10.1152/jn.00183.2009.

[19] Peter Kloppenburg et al. “Neural coding: sparse but on time”. In: *Current Biology* 24.19 (2014), R957–R959.

[20] Rachel I Wilson et al. “Transformation of olfactory representations in the Drosophila antennal lobe”. In: *Science* 303.5656 (2004), pp. 366–370.

[21] Glenn C Turner et al. “Olfactory representations by Drosophila mushroom body neurons”. In: *Journal of neurophysiology* 99.2 (2008), pp. 734–746.

[22] Hannes Rapp et al. “A spiking neural program for sensorimotor control during foraging in flying insects”. In: *Proceedings of the National Academy of Sciences* 117.45 (2020), pp. 28412–28421.

[23] Daniel G Wustenberg et al. “Current-and voltage-clamp recordings and computer simulations of Kenyon cells in the honeybee”. In: *Journal of neurophysiology* 92.4 (2004), pp. 2589–2603.

[24] Collins Assisi et al. “Optimality of sparse olfactory representations is not affected by network plasticity”. In: *PLoS computational biology* 16.2 (2020), e1007461.

[25] Hannes Rapp et al. “Numerical cognition based on precise counting with a single spiking neuron”. In: *IScience* 23.2 (2020).

[26] Toshihide Hige et al. “Heterosynaptic plasticity underlies aversive olfactory learning in Drosophila”. In: *Neuron* 88.5 (2015), pp. 985–998.

[27] Martin Giurfa et al. “Invertebrate learning and memory: fifty years of olfactory conditioning of the proboscis extension response in honeybees”. In: *Learning & memory* 19.2 (2012), pp. 54–66.

[28] Germain U Busto et al. “Olfactory learning in Drosophila”. In: *Physiology* 25.6 (2010), pp. 338–346.

[29] Anna-Maria Jürgensen et al. “Prediction error drives associative learning and conditioned behavior in a spiking model of Drosophila larva”. In: *Iscience* 27.1 (2024).

[30] Rinaldo Betkiewicz et al. “Circuit and cellular mechanisms facilitate the transformation from dense to sparse coding in the insect olfactory system”. In: *eNeuro* 7 (2 Mar. 2020). DOI: 10.1523/ENEURO.0305-18.2020.

[31] A Diamond et al. “Classifying continuous, real-time e-nose sensor data using a bio-inspired spiking network modelled on the insect olfactory system”. In: *Bioinspiration & biomimetics* 11.2 (2016), p. 026002.

[32] Anna-Maria Jürgensen et al. “A neuromorphic model of olfactory processing and sparse coding in the Drosophila larva brain”. In: *Neuromorphic Computing and Engineering* 1.2 (2021), p. 024008.

[33] Michael Schmuker et al. “A neuromorphic network for generic multivariate data classification”. In: *Proceedings of the National Academy of Sciences* 111 (6 Feb. 2014). Using ”on par” machine learning models are necessary to compare the performance of the neuromorphic with its ”ML counterpart”, pp. 2081–2086. DOI: `10.1073/pnas.1303053111`.

[34] Marcel Stimberg et al. “Brian 2, an intuitive and efficient neural simulator”. In: *elife* 8 (2019), e47314.

[35] Robert D Hawkins et al. “Associative learning in invertebrates”. In: *Cold Spring Harbor Perspectives in Biology* 7.5 (2015), a021709.

[36] Thomas Jacob Koickal et al. “Analog VLSI design of an adaptive neuromorphic chip for olfactory systems”. In: *2006 IEEE International Symposium on Circuits and Systems*. IEEE. 2006, 4–pp.

[37] Krishna Persaud et al. “Analysis of discrimination mechanisms in the mammalian olfactory system using a model nose”. In: *Nature* 299.5881 (1982), pp. 352–355.

[38] Giacomo Indiveri et al. “Neuromorphic silicon neuron circuits”. In: *Frontiers in neuroscience* 5 (2011), p. 9202.

[39] Nabil Imam et al. “Rapid online learning and robust recall in a neuromorphic olfactory circuit”. In: *Nature Machine Intelligence* 2.3 (2020), pp. 181–191.

[40] Mike Davies et al. “Loihi: A neuromorphic manycore processor with on-chip learning”. In: *Ieee Micro* 38.1 (2018), pp. 82–99.

[41] Ruben Guerrero-Rivera et al. “Attractor-based pattern classification in a spiking FPGA implementation of the olfactory bulb”. In: *2007 3rd International IEEE/EMBS Conference on Neural Engineering*. IEEE. 2007, pp. 593–599.

[42] Nabil Imam et al. “Implementation of olfactory bulb glomerular-layer computations in a digital neurosynaptic core”. In: *Frontiers in Neuroscience* 6 (2012), p. 83.

[43] Ayon Borthakur et al. “A spike time-dependent online learning algorithm derived from biological olfaction”. In: *Frontiers in neuroscience* 13 (2019), p. 656.

[44] Michael Beyeler et al. “Exploring olfactory sensory networks: simulations and hardware emulation”. In: *2010 Biomedical Circuits and Systems Conference (BioCAS)*. IEEE. 2010, pp. 270–273.

[45] Derek J Hoare et al. “Modeling peripheral olfactory coding in Drosophila larvae”. In: *PLoS One* 6.8 (2011), e22996.

[46] Mehrab N Modi et al. “The Drosophila mushroom body: from architecture to algorithm in a learning circuit”. In: *Annual review of neuroscience* 43 (2020), pp. 465–484.

[47] Baktash Babadi et al. “Sparseness and expansion in sensory representations”. In: *Neuron* 83.5 (2014), pp. 1213–1226.

[48] Katharina Eichler et al. “The complete connectome of a learning and memory centre in an insect brain”. In: *Nature* 548.7666 (2017), pp. 175–182.

[49] David Owald et al. “Activity of defined mushroom body output neurons underlies learned olfactory behavior in Drosophila”. In: *Neuron* 86.2 (2015), pp. 417–427.

[50] Anthony N Burkitt. “A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input”. In: *Biological cybernetics* 95 (2006), pp. 1–19.

[51] John B Furness. “Types of neurons in the enteric nervous system”. In: *Journal of the autonomic nervous system* 81.1-3 (2000), pp. 87–96.

[52] Gordon M Shepherd. *The synaptic organization of the brain*. Oxford university press, 2003.

[53] Santiago Ramon y Cajal et al. “Histologie du système nerveux de l’homme et des vertébrés”. In: *Histologie du système nerveux de l’homme et des vertébrés*. 1955, pp. 2v–2v.

[54] Wulfram Gerstner et al. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[55] Suzana Herculano-Houzel. “The remarkable, yet not extraordinary, human brain as a scaled-up primate brain and its associated cost”. In: *Proceedings of the National Academy of Sciences* 109.supplement_1 (2012), pp. 10661–10668.

[56] Brian S Meldrum. “Glutamate as a neurotransmitter in the brain: review of physiology and pathology”. In: *The Journal of nutrition* 130.4 (2000), 1007S–1015S.

[57] François Python et al. “Immunoreactivity against choline acetyltransferase, γ-aminobutyric acid, histamine, octopamine, and serotonin in the larval chemosensory system of Dosophila melanogaster”. In: *Journal of Comparative Neurology* 453.2 (2002), pp. 157–167.

[58] Akinori Akaike et al. “Excitatory and inhibitory effects of dopamine on neuronal activity of the caudate nucleus neurons in vitro”. In: *Brain research* 418.2 (1987), pp. 262–272.

[59] Robert C Malenka. “Postsynaptic factors control the duration of synaptic enhancement in area CA1 of the hippocampus”. In: *Neuron* 6.1 (1991), pp. 53–60.

[60] Wade G Regehr. “Short-term presynaptic plasticity”. In: *Cold Spring Harbor perspectives in biology* 4.7 (2012), a005702.

[61] Robert S Zucker et al. “Short-term synaptic plasticity”. In: *Annual review of physiology* 64.1 (2002), pp. 355–405.

[62] Robert S Zucker. “Short-term synaptic plasticity”. In: *Annual review of neuroscience* 12.1 (1989), pp. 13–31.

[63] Jon Palacios-Filardo et al. “Neuromodulation of hippocampal long-term synaptic plasticity”. In: *Current opinion in neurobiology* 54 (2019), pp. 37–43.

[64] Luisa Speranza et al. “Dopamine: The neuromodulator of long-term synaptic plasticity, reward and movement control”. In: *Cells* 10.4 (2021), p. 735.

[65] Wolfram Schultz et al. “A neural substrate of prediction and reward”. In: *Science* 275.5306 (1997), pp. 1593–1599.

[66] Jan Benda et al. “A universal model for spike-frequency adaptation”. In: *Neural computation* 15.11 (2003), pp. 2523–2564.

[67] DA Brown et al. “Muscarinic suppression of a novel voltage-sensitive K+ current in a vertebrate neurone”. In: *Nature* 283.5748 (1980), pp. 673–676.

[68] DV Madison et al. “Control of the repetitive discharge of rat CA 1 pyramidal neurones in vitro.” In: *The Journal of physiology* 354.1 (1984), pp. 319–331.

[69] Ilya A Fleidervish et al. “Slow inactivation of Na+ current and slow cumulative spike adaptation in mouse and guinea-pig neocortical neurones in slices.” In: *The Journal of physiology* 493.1 (1996), pp. 83–97.

[70] Jan Benda. “Neural adaptation”. In: *Current Biology* 31 (3 Feb. 2021), R110–R116. DOI: `10.1016/j.cub.2020.11.054`.

[71] Chittotosh Ganguly et al. “Spike frequency adaptation: bridging neural models and neuromorphic applications”. In: *Communications Engineering* 3.1 (2024), p. 22.

[72] Mehdi Adibi et al. “Adaptation improves neural coding efficiency despite increasing correlations in variability”. In: *Journal of Neuroscience* 33.5 (2013), pp. 2108–2120.

[73] Naama Brenner et al. “Adaptive rescaling maximizes information transmission”. In: *Neuron* 26.3 (2000), pp. 695–702.

[74] Stefano Cavallari et al. “Comparison of the dynamics of neural interactions between current-based and conductance-based integrate-and-fire recurrent networks”. In: *Frontiers in neural circuits* 8 (2014), p. 12.

[75] Alan L Hodgkin et al. “A quantitative description of membrane current and its application to conduction and excitation in nerve”. In: *The Journal of physiology* 117.4 (1952), p. 500.

[76] Péter Érdi. “Teaching computational neuroscience”. In: *Cognitive neurodynamics* 9 (2015), pp. 479–485.

[77] Alain Destexhe et al. “The high-conductance state of neocortical neurons in vivo”. In: *Nature reviews neuroscience* 4.9 (2003), pp. 739–751.

[78] LM Lapicque. “Recherches quantitatives sur l’excitation electrique des nerfs”. In: *J Physiol Paris* 9 (1907), pp. 620–635.

[79] Nicolas Brunel et al. “Quantitative investigations of electrical nerve excitation treated as polarization”. In: *Biological Cybernetics* 97.5-6 (2007), p. 341.

[80] Alain Destexhe et al. “Fluctuating synaptic conductances recreate in vivo-like activity in neocortical neurons”. In: *Neuroscience* 107.1 (2001), pp. 13–24.

[81] Yoshinori Aso et al. “The mushroom body of adult Drosophila characterized by GAL4 drivers”. In: *Journal of neurogenetics* 23.1-2 (2009), pp. 156–172.

[82] Ronald L Davis. “Traces of Drosophila memory”. In: *Neuron* 70.1 (2011), pp. 8–19.

[83] Ronald L Davis. “Olfactory learning”. In: *Neuron* 44.1 (2004), pp. 31–48.

[84] Yoshinori Aso et al. “The neuronal architecture of the mushroom body provides a logic for associative learning”. In: *elife* 3 (2014), e04577.

[85] David Marr et al. “A theory of cerebellar cortex”. In: *From the Retina to the Neocortex: Selected Papers of David Marr* (1991), pp. 11–50.

[86] Charles F Stevens. “What the fly’s nose tells the fly’s brain”. In: *Proceedings of the National Academy of Sciences* 112.30 (2015), pp. 9460–9465.

[87] Yoshinori Aso et al. “Mushroom body output neurons encode valence and guide memory-based action selection in Drosophila”. In: *elife* 3 (2014), e04580.

[88] Toshihide Hige et al. “Plasticity-driven individualization of olfactory coding in mushroom body output neurons”. In: *Nature* 526.7572 (2015), pp. 258–262.

[89] Fan Wang et al. “Odorant receptors govern the formation of a precise topographic map”. In: *Cell* 93.1 (1998), pp. 47–60.

[90] Ariane Ramaekers et al. “Glomerular maps without cellular redundancy at successive levels of the Drosophila larval olfactory circuit”. In: *Current biology* 15.11 (2005), pp. 982–992.

[91] Kensaku Mori et al. “The olfactory bulb: coding and processing of odor molecule information”. In: *Science* 286.5440 (1999), pp. 711–715.

[92] Marien De Bruyne et al. “Odor coding in a model olfactory organ: The Drosophila maxillary palp”. In: *Journal of Neuroscience* 19.11 (1999), pp. 4520–4532.

[93] Reinhard F Stocker. “The organization of the chemosensory system in Drosophila melanogaster: a rewiew”. In: *Cell and tissue research* 275 (1994), pp. 3–26.

[94] Michael A Yassa et al. “Pattern separation in the hippocampus”. In: *Trends in neurosciences* 34.10 (2011), pp. 515–525.

[95] Nicolas Y Masse et al. “Olfactory information processing in Drosophila”. In: *Current Biology* 19.16 (2009), R700–R713.

[96] Mehmet Fişek et al. “Stereotyped connectivity and computations in higher-order olfactory neurons”. In: *Nature neuroscience* 17.2 (2014), pp. 280–288.

[97] JW Wang. “Wong AM, Flores J, Vosshall LB, and Axel R”. In: *Two-photon calcium imaging reveals an odor-evoked map of activity in the fly brain. Cell* 112 (2003), pp. 271–282.

[98] Susan E Fahrbach. “Structure of the mushroom bodies of the insect brain”. In: *Annu. Rev. Entomol.* 51 (2006), pp. 209–232.

[99] Rachel I Wilson et al. “Role of GABAergic inhibition in shaping odor-evoked spatiotemporal patterns in the Drosophila antennal lobe”. In: *Journal of Neuroscience* 25.40 (2005), pp. 9069–9079.

[100] RF Stocker et al. “Neuronal architecture of the antennal lobe in Drosophila melanogaster”. In: *Cell and tissue research* 262 (1990), pp. 9–34.

[101] Iori Ito et al. “Olfactory learning and spike timing dependent plasticity”. In: *Communicative & Integrative Biology* 1.2 (2008), pp. 170–171.

[102] Sophie JC Caron et al. “Random convergence of olfactory inputs in the Drosophila mushroom body”. In: *Nature* 497.7447 (2013), pp. 113–117.

[103] Eyal Gruntman et al. “Integration of the olfactory code across dendritic claws of single mushroom body neurons”. In: *Nature neuroscience* 16.12 (2013), pp. 1821–1829.

[104] Xu Liu et al. “The GABAergic anterior paired lateral neuron suppresses and is suppressed by olfactory learning”. In: *Nature neuroscience* 12.1 (2009), pp. 53–59.

[105] Laurence PC Lewis et al. “A higher brain circuit for immediate integration of conflicting sensory information in Drosophila”. In: *Current biology* 25.17 (2015), pp. 2203–2214.

[106] Scott Waddell. “Reinforcement signalling in Drosophila; dopamine does it all after all”. In: *Current opinion in neurobiology* 23.3 (2013), pp. 324–329.

[107] Piergiorgio Strata et al. “Dale’s principle”. In: *Brain research bulletin* 50.5 (1999), pp. 349–350.

[108] Tiago V Maia. “Reinforcement learning, conditioning, and the brain: Successes and challenges”. In: *Cognitive, Affective, & Behavioral Neuroscience* 9.4 (2009), pp. 343–364.

[109] Wolfram Schultz et al. “Neuronal coding of prediction errors”. In: *Annual review of neuroscience* 23.1 (2000), pp. 473–500.

[110] Rescorla Ra. “A theory of Pavlovian conditioning: variations in the effectiveness of reinforcement and nonreinforcement”. In: *Classsical conditioning II: Current research and theory* (1972), pp. 64–99.

[111] Wolfram Schultz. “Dopamine reward prediction error coding”. In: *Dialogues in clinical neuroscience* 18.1 (2016), pp. 23–32.

[112] John Backus. “Can programming be liberated from the von Neumann style? A functional style and its algebra of programs”. In: *Communications of the ACM* 21.8 (1978), pp. 613–641.

[113] Paul A Merolla et al. “A million spiking-neuron integrated circuit with a scalable communication network and interface”. In: *Science* 345.6197 (2014), pp. 668–673.

[114] Paul A Merolla et al. “Expandable networks for neuromorphic chips”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 54.2 (2007), pp. 301–311.

[115] Saber Moradi et al. “A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs)”. In: *IEEE transactions on biomedical circuits and systems* 12.1 (2017), pp. 106–122.

[116] Steve B Furber et al. “The spinnaker project”. In: *Proceedings of the IEEE* 102.5 (2014), pp. 652–665.

[117] Filipp Akopyan et al. “Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip”. In: *IEEE transactions on computer-aided design of integrated circuits and systems* 34.10 (2015), pp. 1537–1557.

[118] Dan F Goodman et al. “The brian simulator”. In: *Frontiers in neuroscience* 3 (2009), p. 643.

[119] Elisabetta Chicca et al. “Neuromorphic electronic circuits for building autonomous cognitive systems”. In: *Proceedings of the IEEE* 102.9 (2014), pp. 1367–1388.

[120] Chiara Bartolozzi et al. “Synaptic dynamics in analog VLSI”. In: *Neural computation* 19.10 (2007), pp. 2581–2603.

[121] Ole Richter et al. “DYNAP-SE2: a scalable multi-core dynamic neuromorphic asynchronous spiking neural network processor”. In: *Neuromorphic Computing and Engineering* 4.1 (2024), p. 014003.

[122] Garrick Orchard et al. “Efficient neuromorphic signal processing with loihi 2”. In: *2021 IEEE Workshop on Signal Processing Systems (SiPS)*. IEEE. 2021, pp. 254–259.

[123] Alexander Vergara et al. “On the performance of gas sensor arrays in open sampling systems using Inhibitory Support Vector Machines”. In: *Sensors and Actuators B: Chemical* 185 (2013), pp. 462–477.

[124] Lava Software Framework. *A software framework for neuromorphic computing*. 2021. URL: `http://lava-nc.org`.

[125] Ruben Guerrero-Rivera et al. “Programmable logic construction kits for hyper-real-time neuronal modeling”. In: *Neural computation* 18.11 (2006), pp. 2651–2679.

[126] David Harris et al. *Digital design and computer architecture*. Morgan Kaufmann, 2010.

[127] Kaveh Akbarzadeh-Sherbaf et al. “A scalable FPGA architecture for randomly connected networks of hodgkin-huxley neurons”. In: *Frontiers in Neuroscience* 12 (2018), p. 342315.

[128] Xilinx. *Floating-Point Operator v7.1 LogiCORE IP Product Guide*. 2020. URL: `https://docs.amd.com/v/u/en-US/pg060-floating-point`.

[129] AMBA ARM. “Axi protocol specification (rev 2.0)”. In: *Availableat http://www. arm. com* (2010).

[130] Alper Yegenoglu et al. *Elephant–open-source tool for the analysis of electrophysiological data sets*. Tech. rep. Computational and Systems Neuroscience, 2015.

[131] Derek J Hoare et al. “Precise and fuzzy coding by olfactory sensory neurons”. In: *Journal of Neuroscience* 28.39 (2008), pp. 9710–9722.

[132] Guangwei Si et al. “Structured odorant response patterns across a complete olfactory receptor neuron population”. In: *Neuron* 101.5 (2019), pp. 950–962.

A
Lava Framework

Lava provides a convenient platform for the development and testing of neuromorphic algorithms on conventional von-Neumann hardware, such as CPUs [124]. It facilitates a prototyping platform before deployment onto specialized neuromorphic processors like the Intel Loihi 1 and Loihi 2 processors. Moreover, Lava can accommodate custom implementations of neuromorphic behavior and to support diverse hardware backends, which can be accessed through their Python libraries.

The architecture of the Lava framework is structured to separate the interface from the behavior of the neuron process, enhancing its compatibility with hardware, which is done by creating a `Process` and its `ProcessModel`. The `Process` Python file defines the specifications of the interface, encompassing input and output characteristics. The `ProcessModel` Python file encapsulates the behavioral aspects of the model, including neuron equations and synaptic dynamics. This framework facilitates the utilization of a uniform interface across various hardware backends, enabling the integration of multiple neurons with distinct specifications but a shared interface, as defined in the `Process`. Consequently, to characterize the neuron (the soma) and its computational operations, a *ProcessModel* named modelCOBALIF.py and a *Process* named ProcessCOBALIF.py were developed, which can be accessed upon request. The design of these models follows the Lava framework.
A `Process` contains three building components, which characterize the connectivity of a `ProcessModel`. It contains two definitions of ports (`InPort`, `OutPort`) that interface with other modules, and a set of interface-specific attributes (`Var`):

- Input ports (`InPort`): input ports connect the current `Process` with other `Processes`.
- Output ports (`Outport`): output ports link the outputs of the current `Process` with other `Processes`.
- Variables (`Var`): each `Process` may have internal variables that can be instantiated through the main.py code, and can only be accessed by the respective `ProcessModel`.

Even synapses require a `ProcessModel` and `Process`. For this model, the `Dense ProcessModel` and `Process` will be used, which essentially translate a connectivity matrix into a Lava-proof module with defined ports. Figure A.1 shows illustrations of two `Process` blocks, each containing two input ports, two variables, and one output port. The connection between the `Process` blocks is generated by another `Process` for synaptic connections, such as the `Dense Process` with a one-to-one connectivity matrix.

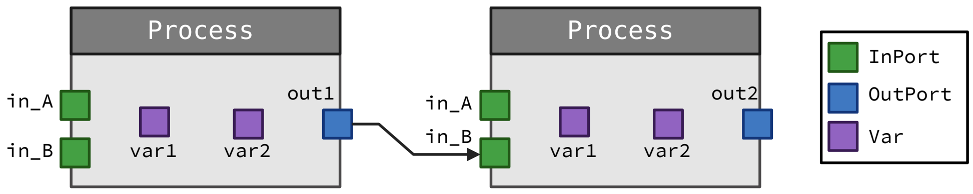


Figure A.1: Fundamental building blocks in the Lava architecture. The figure shows two `Process` blocks, each containing two input ports, two variables, and one output port. The connection between the `Process` blocks is generated by another `Process` for synaptic connections, such as the `Dense` `Process` with a one-to-one connectivity matrix.

Lava employs a standardized protocol for transmitting spikes across layers and processes [124]. Consequently, for this thesis model, an additional script is built to verify that the input data adheres to the specified format, ensuring that only a single spike occurs within each timestep, mirroring the approach adopted in Brian2. This script can be accessed upon request. Input data is supplied through the `Source` module, which generates an array buffer containing input signals for each input neuron (ORN), indicating spike occurrences for each timestep. Each neuron (`Process`) has two input ports for inhibitory and excitatory spike input, and one output port that sends out the spikes to the next `Process` (i.e. via a synapse). During each timestep, only one spike per neuron may occur, which is an intrinsic property - found in the Brian2 simulator too.

B
Model parameters

This appendix serves as an overview of the parameters used in software and digital simulations. Table B.1 includes the parameters of the following neurons: ORNs, PNs, LNs, APL, and KCs. The synaptic parameters and weights are presented in Table B.2 and Table B.3, respectively. Table B.4 and Table B.5 shows the adaptation parameters related to SFA, and the general simulation parameters, respectively. The values are based on the model from Jürgensen et al. in [29].

Table B.1: Neuron parameters

	ORN	**PN**	**LN**	**KC**	**APL**
Membrane Capacitance [pF]	C_m^{ORN}	C_m^{PN}	C_m^{LN}	C_m^{KC}	C_m^{APL}
	100	30	50	30	200
Threshold potential [mV]	V_{th}^{ORN}	V_{th}^{PN}	V_{th}^{LN}	V_{th}^{KC}	V_{th}^{APL}
	-35	-30	-30	-35	-30
Resting potential [mV]	V_r^{ORN}	V_r^{PN}	V_r^{LN}	V_r^{KC}	V_r^{APL}
	-60	-59	-59	-55	-60
Leak conductance [nS]	g_L^{ORN}	g_L^{PN}	g_L^{LN}	g_L^{KC}	g_L^{APL}
	5	2.5	2.5	5	5
Leak potential [mV]	E_L^{ORN}	E_L^{PN}	E_L^{LN}	E_L^{KC}	E_L^{APL}
	-60	-59	-59	-55	-60

Table B.2: Synaptic parameters

Excitatory potential E_E [mV]	0
Inhibitory potential E_I [mV]	-75
Excitatory time constant τ_e [ms]	5
Inhibitory time constant τ_i [ms]	10

Table B.3: Synaptic weights

Input-ORN [nS]	$winputORN$	3
ORN-PN [nS]	$wORNPN$	10
ORN-LN [nS]	$wORNLN$	4
LN-PN [nS]	$wLNPN$	1
PN-KC [nS]	$wPNKC$	1
KC-APL [nS]	$wKCAPL$	20
APL-KC [nS]	$wAPLKC$	50

Table B.4: Adaptation parameters

Adaptation time constant [ms]	τ_{Ia}	1000
Adaptation reversal potential [mV]	E_{Ia}	-90
Increase of SFA conductance ORN [nS]	ORN_{SFA}	0.1
Increase of SFA conductance KC [nS]	KC_{SFA}	0.02

Table B.5: Simulation parameters

Refractory time [ms]	τ_r	2
Time step [ms]	dt	0.1