Real-time bio-hybrid system with spiking neural network

Romain Beaubois

THÈSE PRÉSENTÉE

POUR OBTENIR LE GRADE DE

DOCTEUR
DE L'UNIVERSITÉ DE BORDEAUX

ECOLE DOCTORALE SCIENCES PHYSIQUES ET DE L'INGÉNIEUR

ÉLECTRONIQUE

Par **Romain BEAUBOIS**

Système biohybride temps-réel avec Spiking Neural Network

Real-time biohybrid system with Spiking Neural Network

Sous la direction de : **Timothée LÉVI**
Co-directeur : **Pascal BRANCHEREAU**
Co-directeur : **Yoshiho IKEUCHI**

Soutenue le 11 décembre 2023

Membres du jury présidé par Dominique DALLET :

M. Timothée LÉVI	Professeur des Universités	Université de Bordeaux	Directeur de thèse
M. Pascal BRANCHEREAU	Professeur des Universités	Université de Bordeaux	Co-directeur de thèse
M. Yoshiho IKEUCHI	Maître de conférences	Université de Tokyo	Co-directeur de thèse
M. Benoît MIRAMOND	Professeur des Universités	Université de Côte d'Azur	Rapporteur
M. Blaise YVERT	Directeur de Recherche	INSERM Grenoble	Rapporteur
M. Paolo BONIFAZI	Maître de conférences	Université d'Ikerbasque	Examinateur
M. Dominique DALLET	Professeur des Universités	Bordeaux INP	Examinateur
Mme. Noëlle LEWIS	Professeure des Universités	Université de Bordeaux	Examinatrice

This thesis was conducted at

IMS Laboratory, BioElectronics group
CNRS UMR-5218
University of Bordeaux
351 Cours de la Libération, 33405 Talence CEDEX
France

in international collaboration including a one-year stay at

Ikeuchi Lab
LIMMS-CNRS, Institute of Industrial Science
The University of Tokyo
4-6-1 Komaba, Meguro-ku
153-8505 Tokyo
Japan

Acknowledgments

The completion of the present work is not solely attributed to my efforts but is rather a culmination of close contributions and collaborations with individuals who played pivotal roles in enabling this multidisciplinary endeavor to thrive and reveal its full potential. Hence, it is imperative to duly acknowledge and attribute credit where credit is due.

First and foremost, I extend my heartfelt gratitude to Timothée Lévi, my dedicated director who demonstrated exceptional skill in navigating the delicate journey of doctoral research through the vast and challenging realm of research. He, more than anyone, guided, enlightened, supported and inspired me throughout this long and intricate journey with passion and ambition. Needless to say, our long-standing collaboration undoubtedly influenced many of the choices I made and played a significant role in shaping the researcher I have become, none of which I would regret.

In addition, my profound gratitude extends to my co-directors, Pascal Branchereau and Yoshiho Ikeuchi, whose steadfast support and invaluable insights have significantly enriched my journey in the realm of neurosciences. I express particular gratitude for Yoshiho Ikeuchi's mentorship, guidance, and expertise during my numerous stays in his laboratory over the several years of our collaboration.

I would like to express my sincere gratitude to Benoît Miramond and Blaise Yvert, who dedicated their time to meticulously review and provide invaluable feedback on the manuscript. Their insightful comments and constructive criticism have played a crucial role in shaping the final version of this work.

I extend my gratitude to the additional jury members, Paolo Bonifazi, Noëlle Lewis and Dominique Dallet, for their valuable contributions to the evaluation of this work through their expertise and thoughtful insights.

I am also giving a word of appreciations to all the members of our research group at IMS and more particularly to the TIPS team for their valuable insights and support. I express my deepest gratitude to inseparable pair, Pierre-Marie Faure and Jérémy Cheslet. Their unwavering support, shared laughter, and enduring companionship have been a source of strength and joy throughout this journey. I am especially thankful to Jérémy Cheslet for the considerable contribution to this work as an intern.

Furthermore, I would like to acknowledge the University of Bordeaux through its different programs that supported me during all my doctoral thesis nationally and internationally. Also expressing gratitude to The University of Tokyo through the Institute of Industrial Science for providing an exceptional academic environment and invaluable resources, which have greatly contributed to the success of this research endeavor. Indeed, my gratitude extends to all members of the LIMMS for their unconditional support and warm atmosphere that allowed for this international mobility to unfold seamlessly while providing enriching opportunities.

Without a doubt, I want to convey my deepest appreciation to our collaborators in Italy and Japan whose contributions significantly enhanced the quality and depth of this research.

I would like to express my sincere appreciation to all members of the Ikeuchi. Additionally, I would like to extend special thanks to Tomoya Duenki whose friendship and contribution considerably participated in the success of this research, Siu Yu Angela Chow whose cheerful attitudes infused the lab with positivity as well as Tatsuya Osaki for his valuable expertise.

In addition, I express my gratitude to all the members of our collaborating team in Italy

for their valuable work, expertise and insights. Moving from the collective to the individual, I would like to thank Michela Chiappalone, Mattia Di Florio and Marta Carè for significantly enhancing our understanding and experiments with their knowledge, passion and dedication.

Undoubtedly, I greatly thank my parents, grandparents, sisters and friends who always believed in me and supported me in all circumstances.

Finally, I want to convey my profound appreciation to Haruka Ono for her unwavering support, love and encouragement. Her presence has been a constant source of strength, making every challenge more manageable and every success more meaningful.

Resumé

La caractérisation et la modélisation des réseaux neuronaux biologiques ont émergé comme un domaine permettant des avancées significatives dans notre compréhension des fonctions cérébrales et des pathologies qui y sont liées.

À ce jour, les traitements pharmacologiques des troubles neurologiques restent limités, ce qui pousse à explorer des approches alternatives prometteuses telles que l'électroceutique. Les recherches récentes en bioélectronique et en ingénierie neuromorphique ont conduit à la conception d'une nouvelle génération de neuroprothèses pour la réhabilitation du cerveau.

Toutefois, leur développement complet nécessite une compréhension et une expertise plus approfondies de l'interaction biohybride. Ici, ce travail de thèse présente un nouveau réseau de neurones biomimétique temps réel à la fois abordable, flexible et accessible pour la réalisation d'expériences bio-hybrides et l'émulation en temps réel.

Ce réseau biomimétique permet d'étudier et de reproduire la dynamique de réseaux de neurones détaillés sur le plan biophysique tout en promouvant une flexibilité et facilité d'utilisation. Il démontre la faisabilité d'expériences biohybrides utilisant des interfaces biophysiques standards et diverses cellules biologiques, ainsi que l'émulation en temps réel de modèles complexes. Le système mis au point permet de réaliser des expériences biohybrides ainsi que l'émulation en temps réel de réseaux de neurones.

Le système développé constitue une étape essentielle vers le développement de neuroprothèses neuromorphiques pour les thérapies bioélectriques comme l'électroceutique. Elle permet également la communication avec des réseaux de neurones biologiques sur une échelle de temps similaire, facilitée par un système en temps réel embarqué, facile à utiliser et accessible. Le dispositif en temps réel développé démontre son potentiel dans des applications pratiques et expériences biohybrides.

Mots clés: *SoC FPGA, Réseau de neurones, Bio-hybride, Maladies neurodégénératives, Ingénieurie neuromorphique*

Abstract

Characterization and modeling of biological neural networks has emerged as a field driving significant advancements in our understanding of brain function and related pathologies.

As of today, pharmacological treatments for neurological disorders remain limited, pushing the exploration of promising alternative approaches such as electroceutics. Recent research in bioelectronics and neuromorphic engineering have led to the design of the new generation of neuroprostheses for brain repair.

However, its complete development requires deeper understanding and expertise in biohybrid interaction. Here, this thesis work shows a novel real-time, biomimetic, cost-effective and user-friendly neural network for bio-hybrid experiments and real-time emulation.

This thesis work allows investigation and reproduction of biophysically detailed neural network dynamics while promoting cost-efficiency, flexibility and ease of use. It showcases the feasibility of conducting biohybrid experiments using standard biophysical interfaces and various biological cells as well as real-time emulation of complex models.

The system developed in this work is anticipated to be a step towards developing neuromorphic-based neuroprostheses for bioelectrical therapeutics by enabling communication with biological networks on a similar timescale, facilitated by an easy-to-use and accessible embedded real-time system. The real-time device developed further enhances its potential for practical applications in biohybrid experiments.

Keywords: *SoC FPGA, Spiking Neural Network, Biohybrid, Neurodegenerative diseases, Neuromorphic engineering*

Glossary

GLOSSARY - Common

AD Alzheimer's Disease.

Adex Adaptive Exponential Integrate and Fire.

AIS Axon Initial Segment.

ALS Amyotrophic Lateral Sclerosis.

ANN Artificial Neural Network.

BCI Brain Computer Interface.

BMI Brain Machine Interface.

BNN Biological Neural Network.

CPG Central Pattern Generator.

EIF Exponential Integrate and Fire.

FS Fast Spiking.

HD Huntington's Disease.

HD-MEA High Density MicroElectrode Array.

HH Hodgkin-Huxley.

IB Intrinsic Bursting.

IBI InterBurst-Interval.

IF Integrate-and-Fire.

iPSCs Induced Pluripotent Stem Cells.

ISI InterSpike-Interval.

IZ Izhikevich.

LIF Leaky Integrate-and-Fire.

LTS Low-Threshold Spiking.

MEA MicroElectrode Array.

MFR Mean Firing Rate.

NEA NanoElectrode Array.

OOC Organ-On-Chip.

PD Parkinson's Disease.

PSCs Pluripotent Stem Cells.

RS Regular Spiking.

SNN Spiking Neural Network.

STDP Spike Timing Dependent Plasticity.

STP Short-Term synaptic plasticity.

GLOSSARY - Computer science

AMBA Advanced Microcontroller Bus Architecture.

APU Application Processing Unit.

ASIC Application-Specific Integrated Circuit.

AXI Advanced eXtensible Interface.

BD Buffer Descriptor.

CPU Central Processing Unit.

DAC Digital-to-Analog Converter.

DMA Direct Memory Access.

DMD Digital Micromirror Device.

DSP Digital Signal Processor.

EMIO Extended Multiplexed Input/Output.

FF Flip-Flop.

FPGA Field Programmable Gate Array.

FSBL First Stage BootLoader.

GEM Gigabit Ethernet MAC.

GIC Generic Interrupt Controller.

GPIO General Purpose Input/Output.

GPU Graphics Processing Unit.

GUI Graphical User Interface.

HLS High Level Synthesis.

IOP Input Output Peripheral.

IR Infrared.

LUT Look Up Table.

MIO Multiplexed Input/Output.

PL Programmable Logic.

PMOD Peripheral MODule interface.

PS Processing System.

RPU Real-Time Processing Unit.

RTL Register Transfer Level.

SGMII Serial Gigabit Media-Independent Interface.

SIMD Single-Instruction Multiple-Data.

SoC System on Chip.

SOM System-on-Module.

SPI Serial Peripheral Interface.

UART Universal Asynchronous Receiver/Transmitter.

USB Universal Serial Bus.

VHDL VHSIC Hardware Description Language.

Table of contents

Table of Contents

Acknowledgments **3**

Resumé **6**

Abstract **8**

Glossary **10**

Table of contents **15**

Introduction **19**

1 Capturing biology in faithful models **23**
1.1 Introduction . . . 24
1.2 Nervous system morphology . . . 24
1.2.1 Neuron . . . 25
1.2.2 Soma . . . 25
1.2.3 Dendrite . . . 26
1.2.4 Axon . . . 26
1.2.5 Plasma membrane . . . 26
1.2.6 Synapses . . . 27
1.2.7 Action potential . . . 27
1.2.8 Neural coding . . . 28
1.3 Neurological disorders . . . 29
1.3.1 Neurodegenerative diseases . . . 30
1.3.2 Existing treatments . . . 31
1.3.3 Innovative alternative treatments . . . 31
1.4 Biological models . . . 33
1.4.1 Modeling human body . . . 33
1.4.2 In-vitro 2D cultures . . . 33
1.4.3 In-vitro 3D cultures . . . 34
1.4.4 In-vivo experiments . . . 34
1.4.5 Data acquisition . . . 35
1.4.6 Model Interactions . . . 37
1.5 Artificial modeling . . . 37
1.5.1 Bio-inspired and biomimetic approaches . . . 37
1.5.2 Neuron models . . . 38
1.5.3 Spontaneous activity . . . 40
1.5.4 Single compartment modeling . . . 41
1.5.5 Multicompartmental modeling . . . 42
1.5.6 Synapse models . . . 44
1.5.7 Neural networks . . . 46
1.6 Numerical system solving for artificial neuron modeling . . . 47
1.6.1 Numeric solvers . . . 47
1.6.2 Single compartment modeling . . . 47
1.6.3 Multicompartmental modeling . . . 48
1.7 Summary . . . 51

2 Introducing the AMD Xilinx SOM K26 **52**
2.1 Introduction . . . 53
2.2 Technological context . . . 53
2.2.1 State of the art of biomimetic models . . . 54
2.2.2 Follow-up on HH FPGA implementations . . . 55
2.2.3 Selected Targets Overview . . . 55
2.3 Communication Protocols and Interfaces . . . 57
2.3.1 AXI Protocol: Efficient SoC Interconnect Communication . . . 57
2.3.2 USB: The Universal Standard for Device Integration . . . 59
2.3.3 Ethernet: Connecting Devices in Networks . . . 59
2.3.4 UART: Basic Device Communication . . . 60
2.3.5 SPI: Simplified Device Connection . . . 61
2.3.6 Wi-Fi: Wireless Device Connectivity . . . 61
2.3.7 PMOD: Standard Device Interfacing . . . 62
2.4 Architecture and components of the SOM K26 . . . 63
2.4.1 Application Processing Unit (APU) . . . 64
2.4.2 Real-Time Processing Unit (RPU) . . . 64
2.4.3 Graphics Processing Unit (GPU) . . . 64
2.4.4 Memory . . . 65
2.4.5 Connectivity . . . 66
2.4.6 System functions and management . . . 66
2.5 Processing System (PS): Embedded Processors within SoC FPGA . . . 68
2.5.1 Bare Metal: Low-Level Hardware Programming . . . 68
2.5.2 FreeRTOS: Real-time operating system . . . 69
2.5.3 Linux: Versatile Embedded System Platform . . . 69
2.6 Programmable Logic (PL): FPGA Technology within SoC FPGA . . . 70
2.6.1 Data encoding . . . 70
2.6.2 High Level Synthesis tools . . . 71
2.6.3 Memory . . . 72
2.6.4 Arithmetical and mathematical operations . . . 73
2.7 Summary . . . 75

3 Toward a flexible real-time biomimetic SNN on SOM K26 **76**
3.1 Introduction . . . 77
3.2 Overview . . . 77
3.3 Hardware: Computation Core . . . 78
3.3.1 Ion channels states . . . 79
3.3.2 Single compartment neurons . . . 81
3.3.3 Multicompartmental neurons . . . 83
3.3.4 Synapses . . . 86
3.3.5 Setup . . . 89
3.4 Hardware: Monitoring . . . 90
3.4.1 Spikes and waves *via* DMA . . . 90
3.4.2 Spikes *via* PMOD ESP32 . . . 91
3.4.3 Waves *via* PMOD DA4 . . . 91
3.4.4 External stimulation . . . 92
3.5 Software: setup, control and monitoring by C++ application . . . 92
3.5.1 Structure and organization . . . 93
3.5.2 Software configuration . . . 94
3.5.3 Hardware configuration . . . 94

3.5.4 Monitoring . . . 95
3.6 Software: configuration and monitoring by Python scripts . . . 96
3.6.1 Configuration . . . 96
3.6.2 Emulation . . . 97
3.6.3 Monitoring . . . 98
3.7 Performances . . . 98
3.7.1 Resource utilization . . . 99
3.7.2 Power . . . 100
3.7.3 Latency . . . 101
3.8 Summary . . . 103

4 Applications and biohybrid experiments 104
4.1 Introduction . . . 105
4.2 Real-time emulation of biophysically detailed neurons . . . 106
4.2.1 Spontaneously spiking single compartment neurons . . . 106
4.2.2 Multicompartmental motor neurons to study ALS . . . 107
4.3 Real-time emulation of complex neuronal structures . . . 111
4.3.1 Interconnection of human cerebral organoids . . . 111
4.3.2 Artificial modeling . . . 112
4.3.3 Real-time emulation . . . 113
4.3.4 Mimic drug treatments . . . 115
4.4 Open-loop biomimetic in-vivo stimulation . . . 116
4.4.1 Electroceutical approach for post-stroke rehabilitation . . . 117
4.4.2 Intermediate version of BiœmuS . . . 117
4.4.3 Experimental setup and protocol . . . 118
4.4.4 Results . . . 119
4.4.5 Discussion . . . 120
4.5 Closed-loop biohybrid spinal cord-brain interaction . . . 120
4.5.1 Snake robot controlled by biomimetic Central Pattern Generators . . . 120
4.5.2 Alternative version of BiœmuS . . . 121
4.5.3 Experimental setup and protocol . . . 122
4.5.4 Results . . . 125
4.5.5 Discussion . . . 126
4.6 Closed-loop biomimetic in-vitro stimulation on high resolution MEA . . . 127
4.6.1 Toward biological intelligence using cortical connectoid . . . 127
4.6.2 Experimental setup and protocol . . . 127
4.6.3 Results . . . 130
4.6.4 Discussion . . . 133
4.7 Summary . . . 134

Conclusion 135

Publications 139

Bibliography 141

Introduction

Millions of people worldwide are affected by neurological disorders that strongly impair their cognitive and/or motor functions [Organization et al., 2020]. An increasing number of technologies and solutions are currently proposed for the treatments of these diseases, whereas being limited to curbing the progress or managing symptoms in most cases [Chin and Vora, 2014, French et al., 2016].

Aside from medical treatment through chemical processes, artificial devices are developed to improve the quality of life of individuals. To bring neuroprosthesis into realization, the behavior of biological neurons as well as its connection and interaction with artificial neural networks must be considered. To this end, investigation of the interaction of neuronal cell assemblies is required to understand and reproduce a specific behavior driven by intrinsic spontaneous activity. Additionally, long-term replacement of damaged brain areas with artificial devices implies understanding of their neurophysiological behaviors.

In this context, new therapeutic approaches and technologies are needed both to promote cell survival and regeneration of local circuits [Farina et al., 2021] and restore long distance communication between disconnected brain regions and circuits [Bouton et al., 2016]. Thus, characterization and modeling of biological neural networks [Panuccio et al., 2018, Semprini et al., 2018] is crucial to develop a new generation of neuroprostheses that mimics biological dynamics and provide adaptive stimulation at biological time scale based on the principle of electroceutics [Famm et al., 2013, Reardon, 2014].

Thanks to the new neuromorphic platforms, performing bio-hybrid experiments is becoming more and more relevant not only for the development of neuromorphic biomedical devices [Famm et al., 2013, Reardon, 2014], but also to elucidate the mechanisms of information processing in the nervous system. Recently, major progress has been made in the field of neuroprostheses [Panuccio et al., 2018, Semprini et al., 2018] so as neuromorphic devices are now capable of receiving and processing input while locally or remotely delivering their output either through electrical, chemical or optogenetic stimulation [Christensen et al., 2022].

However, real-time stimulation and processing of biological data using biomimetic Spiking Neural Network (SNN) is still quite rare [Ambroise et al., 2013, Xu et al., 2018, Buccelli et al., 2019, Mosbacher et al., 2020]. Furthermore, to improve temporal accuracy of the stimulation, complex neuron model should be implemented in the SNN [Sharifshazileh et al., 2021].

To perform bi-directional bio-hybrid experiments and develop bioelectrical therapeutic solutions for health care like electroceutic [Famm et al., 2013, Reardon, 2014, Di Florio et al., 2023], real-time bio-physics interface and SNN processing are mandatory to ensure interaction at biological time scale [Sharifshazileh et al., 2021, Corradi and Indiveri, 2015]. Most of current solutions for biomimetic SNN simulations are software-based such as NEURON [Hines and Carnevale, 2001], NEST [Gewaltig and Diesmann, 2007] or Brian2 [Stimberg et al., 2019] tools and show significantly high computation time, especially for complex neuron model with synaptic plasticity. Hence, these latter are not suited for real-time emulation at millisecond time step [Van Albada et al., 2018] contrary to hardware-based SNNs. Another benefit of hardware-based SNNs is the ability to perform massive parallel simulations to explore space parameters of neuron models.

In the neuromorphic engineering research, SNNs are designed using two distinct approaches: bioinspired or biomimetic. The former is widely used for applications such as computation and artificial intelligence [Tavanaei et al., 2019] using accelerated time simulation of simple neuron

model. The latter uses complex neuron model operating at biological timescale to simulate neural network dynamics or/and performing bio-hybrid experiments.

Hardware-based SNNs are analog or digital. Analog SNN systems [Donati et al., 2019] show lower power consumption than digital SNNs [Davidson and Furber, 2021]. In contrast, digital SNNs are more flexible thus more suited for prototyping while showing overall quicker design time hence constituting the best choice for preliminary experiments and design of new generation of neuroprosthetic. The prominent SNNs hardware platforms are Merolla [Merolla et al., 2014], BrainScaleS-2 [Pehle et al., 2022], SpiNNaker [Painkras et al., 2013] and Loihi [Davies et al., 2018]. While some of these systems present mobile versions like [Stradmann et al., 2022] for BrainScaleS-2, they often are not suited for embedded applications.

This thesis focuses on the design of a real-time bio-hybrid platform capable of biomimetic Spiking Neural Network emulation for the study of neurological disorders through real-time emulation and hybridization. Benefiting from a flexible, real-time and biomimetic architecture, the system developed is intended to be used a tool for neuroscientists to predict and estimate biophysically detailed models efficiently through emulation. More importantly, this system is intended easily integrate biohybrid closed-loop system to explore the electroceutic approach, and hopefully contribute to the development of neuroprostheses.

The BioElectronics group, affiliated with the IMS Laboratory at CNRS UMR5218 and the University of Bordeaux, primarily specializes in the field of analog neuromimetic integrated circuits and hybrid neural-silicon systems. One of the key research areas revolves around the creation of innovative instrumentation tools designed for the exploration of the central nervous system through model emulation and hybridization. The group has been working on artificial modeling and hybridization using analog circuits for more than 20 years [Le Masson et al., 2002], then moved toward digital implementation for the past 10 years [Ambroise et al., 2013]. This thesis marks the third iteration of the group's research on the digital implementation of neural networks on FPGA, building upon its latest work [Khoyratee et al., 2019].

Collaborations with other teams played a crucial role in the fulfillment of this work, especially for successfully conducting the biohybrid experiments. The primary collaboration involved the Institute of Industrial Science (IIS) at The University of Tokyo in Japan, which included a one-year stay at the Ikeuchi Lab, also affiliated with the LIMMS international research unit operating jointly with France and Japan. The Ikeuchi Lab has been collaborating with the team since 2017 and is specialized in the creation of functional neuronal circuits using brain organoids generated from human iPS cells. Most notably, it developed a unique method to create connectoids, which are neural circuit tissues made of brain organoids connected together, allowing investigation of the interaction between distant regions of the brains. The second collaborator for this work was a team from the Istituto Italiano di Tecnologia (IIT) of Genoa in Italy as part of the of the "NEUROHYSTIM: A NEUROHYbrid system to drive intracortical microSTIMulation in neuronal networks in vivo" project. The Italian team has experience and expertise in in-vivo experiments, neurostimulation and analysis of electrophysiological signals [Bologna et al., 2010, Bonifazi et al., 2013, Buccelli et al., 2019].

In this thesis manuscript organized in four chapters, the context and elementary knowledge required for a consistent understanding will be provided as well as the methods that led to the design of a real-time biohybrid system called BiœmuS. The capabilities of the real-time biomimetic SNN BiœmuS to emulate independent neurons and fully connected networks will then be presented, showcasing a system integration promoting versatility and ease of use high-

lights by biohyrid experiments. Intermediate and alternative versions of the system, preliminary work, prototypes and ways of improvement will be shown.

The first chapter aims to introduce the context of the thesis and highlight its relation to the problematic by introducing the notion of model and its importance in scientific studies. Starting with a basic introduction to neurosciences, more specifically to the nervous system morphology and the neurological disorders that can affect it, the use of a model will be justified. After presenting the biological and artificial models along with their benefits and limits, the implementation of artificial models on numerical systems will be detailed.

The second chapter dives in the computer science domain by introducing the platform selected and its main characteristics. The technological context that supported the choice of the platform will be presented, followed by a basic introduction to the essential communication protocols and interfaces involved in the system. Then, a detailed explanation of the characteristics and processes operating in the two main parts of the platform will be given.

The third chapter is dedicated to the development steps of a flexible real-time biomimetic design on the platform previously presented. Firstly, the hardware architecture of the system constituting the core of the real-time system will be described. Then, an explanation of the different software layers developed to interact with the hardware will be provided. Finally, the performances of the system will be discussed.

The fourth chapter focuses on the applications and experiments conducted with the system developed. To begin with, applications of the system a real-emulator targeting independents neurons to large network models will be presented. Afterward, biohybrid experiments conducted in international collaborations targeting various biological cultures will be showcased.

1 Capturing biology in faithful models

1.1 Introduction

Throughout history, Nature has been a major source of inspiration for solving the complex human problems. Bio-inspired inventions, literally inventions that were thought and inspired by nature, are all around us.

From the small things such as Velcro tape inspired by bur fruits hooks, to our greatest achievements as train and planes designs inspired by birds beak and wings, the constantly evolving and adapting nature always hinted us new solutions [Shu et al., 2011]. Needless to say that one of its most mysterious creation that is human brain is no exception.

Human brain is estimated to show an enormous computation power of 1 exaFLOPS, a comparable specification to a supercomputer [Smirnova et al., 2023]. Its memory capacity is estimated to 2.5 petabytes. Extremely high specifications powered by only about 20 W, making of human brain one of the most powerful computation unit in the world [Smirnova et al., 2023]. Highly complex and still partly understood, its tremendous computation power and low energy consumption inspired notably the now widely used artificial neural networks found in AI and cryptocurrencies blockchains [Krogh, 2008].

Human brain and the nervous system are essential elements for communication and coordination in the human body through neurons and synapses. Hence, malfunction in these systems significantly impacts on body functions. Unfortunately, diseases that target the nervous system and human brain are affecting millions of people worldwide [Organization et al., 2020] and show limited treatments that only manage the symptoms and attempt to curb the progression. Thus, it is essential to understand the mechanism governing human brain and nervous system [Chin and Vora, 2014, French et al., 2016].

An important part of research relies on the use of models to predict and investigate processes, biology is no exception. Biological models that reproduce biological processes have been widely used to study human body. With the growth of computer power over the years, artificial models that rely on computer simulations invested a large part of the studies.

This raises the question of faithfulness and utility of a model, in other words, how much a model can be trusted and what are the important criteria characterizing a faithful model. This section will elaborate that subject with the specific case of the modeling of human brain and its nervous system.

The following chapter aims to introduce the context of the thesis and highlight its relation to the problematic.

1.2 Nervous system morphology

The nervous system is an organ of the human being responsible for communication and coordination between the different regions of the body.

It is a complex structure that can be separated in two main parts: the central nervous system constituted of the brain and spinal cord and the peripheral nervous system including nerves that run throughout the whole body. The central nervous system allows reception, processing and sending of sensory information. It is also in charge of voluntary functions such as speaking, walking and breathing.

The peripheral nervous system, for its part, transmits the nervous impulsion generated by the central nervous system to the muscles and organs. It is also in charge of forwarding sensory information to the central nervous system.

It includes neurons that are cells specialized in electrical signal processing and from varying shapes and electrochemical properties depending on its role [Cooper, 2011] as well as other cells like glial cells. Three types of neurons are shown in Figure 1.1.

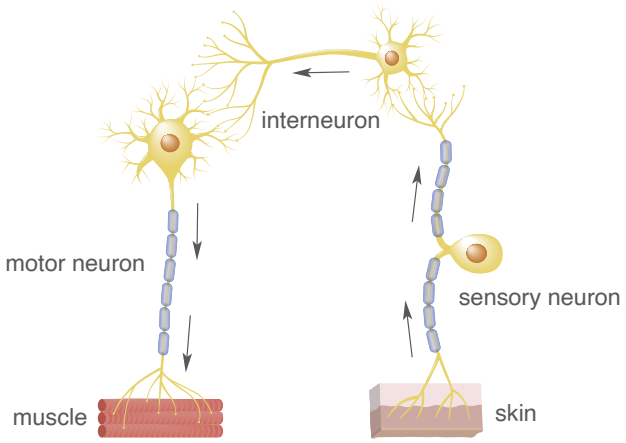


Figure 1.1: Types of neurons found in the nervous system. The interneurons connect neurons together, the motor neurons connect to the muscle tissues and the sensory neurons convey the sensory information coming from body organs.

1.2.1 Neuron

Neurons are key elements of the communication in the nervous system by being responsible for the transmission of the electrical impulsion to the body. This electrical impulsion is called an action potential. Neurons are estimated to be about 120 billions in human brain [Herculano-Houzel, 2009] distributed in about 101 billions in the cerebellum [Andersen et al., 1992] and 21 to 26 billions in cerebral cortex [Pelvig et al., 2008].

A neuron is composed of a cellular body (soma), axon and dendrites (see Figure 1.2). Neurons average size is estimated to about 100 µm in diameter in humans [Cooper, 2011]. The cellular body is the command center that produces the energy needed for nervous impulsion transmission.

The axon is a long cable that propagates and regenerates the electrical impulsion (action potential), while the dendrites are short branches allowing reception of impulsion from other neurons and transmission of this impulsion towards the initial segment located on the axon. The transmission between neurons is carried out by synapses that mainly connect axons to dendrites and to the soma.

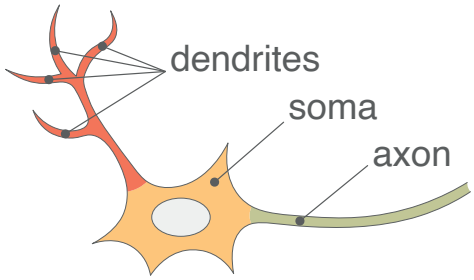


Figure 1.2: Simplified schematic of a neuron showing its main elements.

1.2.2 Soma

The soma (or cell body) of the neuron is the part where the DNA of the neuron is stored, and the proteins required for its functioning produced. Depending on its position in the body, its shape and size can vary. This is also one of the part of the cell where the neurotransmitters

released upon reception of an action potential are received, constituting a unit for processing neuronal information.

1.2.3 Dendrite

Dendrites are extensions organized in leaf-like structures connected to the soma that allow reception of nervous impulsion from other neurons. The diameter is not constant as it decreases from primary to secondary to tertiary dendrites. They have a similar behavior to antenna by receiving and transmitting the electrical impulsion from the others neuron to the cellular body of the neuron. The morphology of dendrites varies and is linked to mechanism of synaptic plasticity [Forrest et al., 2018].

1.2.4 Axon

Axon is a thread-like extension of the neuron having a constant diameter that carries the action potential. Usually, each neuron grows only one axon that extent over a large distance compared to the soma [Kandel et al., 2000]. The electrical impulsion is then transmitted to other neurons or muscles or organs.

In an analogy to electrical circuits, the axon can be seen as a cable. Similar to dendrites, the morphology of the axon vary greatly in length from one another. However, all axons have main regions including the axon hillock, the Axon Initial Segment (AIS) that is around 30 µm long where the action potential originates and where voltage-dependent Na^+ and K^+ channels are concentrated to generate the action potential. It also includes the rest of the axon, the axon telodendria and the terminal part where synapses contain axon terminals going up in total to up to 1 meter.

To enhance the efficiency and speed of signal transmission, some axons are enveloped by a protective sheath called myelin that is generated by oligodendrocytes in the central nervous system and Schwann cells in the periphery. Myelin acts as an insulating layer, allowing electrical impulses to propagate at significantly higher speed in axons thanks to a saltatory conduction.

1.2.5 Plasma membrane

The plasma membrane is a thin layer that can be found in all cells that separates its interior from the outside environment. It is responsible for the regulation of the exchanges between a cell and its environment. Plasma membrane contains specific channels and transporters such as the Na/K/ATPase pump notably enables neurons to generate and propagate ion currents through ions channels that exchange ions between the extracellular and intracellular environments (see Figure 1.3 for the Na/K/ATPase pump).

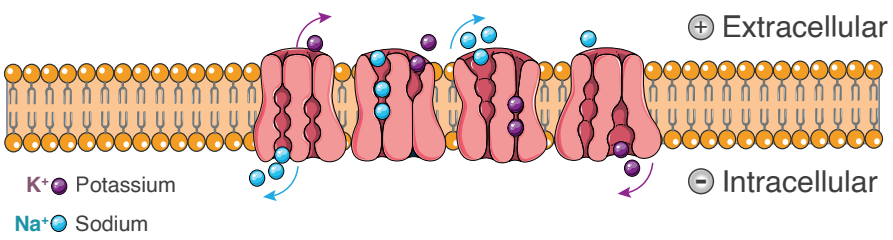


Figure 1.3: Ion channels Na/K/ATPase pump on the membrane axon. Pumps exchange ions between the extracellular and intracellular to propagate the action potential. The Na/K/ATPase pump hydrolyzes an ATP molecule to allow 3 Na+ ions (blue) to exit and 2 K+ ions (purple) to enter. This pump exerts an electrogenic effect by keeping fewer Na^+ in/out and more K^+ in/out, constituting the basis of membrane potential maintenance.

It is constituted of phospholipids, proteins and glycoproteins organized as a semi-permeable barrier allowing proteins to enter the cell while blocking other substances.

1.2.6 Synapses

A synapse is a structure allowing connection between neurons. It connects the axon of a neuron to either the dendrites, axon or soma of another neuron, muscles or organs depending on the type of neuron. It permits transmission of the signal from a neuron to another cell, thus enabling propagation of the signal in the whole body from neuron to its target. The neuron sending its axon is called the presynaptic neuron while the neuron receiving the signal is called postsynaptic neuron. Synapses can either be electrical or chemical based on the signal transmission method.

In a chemical synapse, the electrical impulsion of the presynaptic neuron is converted into the release of chemical called a neurotransmitter. The neurotransmitter binds to the specific receptors found in the plasma membrane of the postsynaptic cell. Synaptic receptors create signal transduction in the postsynaptic cell that can result in complex effects from inhibition (GABAAR, GlyR, ...) to excitation (NMDAR, AMPAR, ...).

Electrical synapses use a structure called gap junctions that are connected channels capable of passing electric current that induce a voltage change in the postsynaptic cell. While the electrical synapse has less amplitude effects on the post synaptic cell, transmission is much faster.

Another special feature of synapses is their plasticity that make a synapse more likely to trigger another action potential faster upon spike reception. The synapse is then said to possess a reinforcing weight. [Hebb, 2005] proposed a theory stating that "When an axon of cell A is close enough to excite B and repeatedly or persistently, a growth process or metabolic change occurs in one or both cells, so that the effectiveness of A, as the cell activating B, is enhanced".

1.2.7 Action potential

An action potential, or spike corresponds to an abrupt change of the membrane potential when the membrane potential of the neuron rapidly rises above 0 mV (change of membrane polarity) and falls, thus allowing the electrical message to propagate along the axon. The propagation is performed through a potential difference between the extra and intracellular environments generated by ion channels that exchange Na^+ and K^+ ions between the environments then creating a potential difference as shown in Figure 1.3. The generation of an action potential shows 4 states: resting, depolarization, repolarization and refractory period (see Figure 1.4).

Resting state. The resting state is characterized by a constant resting potential explained by a disparate distribution of charges and ionic species between the extracellular and intracellular media. The intracellular medium has an excess of K^+ ions and deficit in Na^+ ions, compared to the extracellular medium (see Figure 1.3,1.4). This state is maintained in the absence of stimulation of the cell. The value of the potential depends on the neuron family but often corresponds to around -70 mV. It is explained by an equilibrium point between the equilibrium potential of Na^+ ions ($\sim$ 55 mV) and the equilibrium potential of K^+ ions ($\sim$ -75 mV). These two ionic species being the key players in the generation of action potentials that is closely dependent on the Na/K ATPase pump.

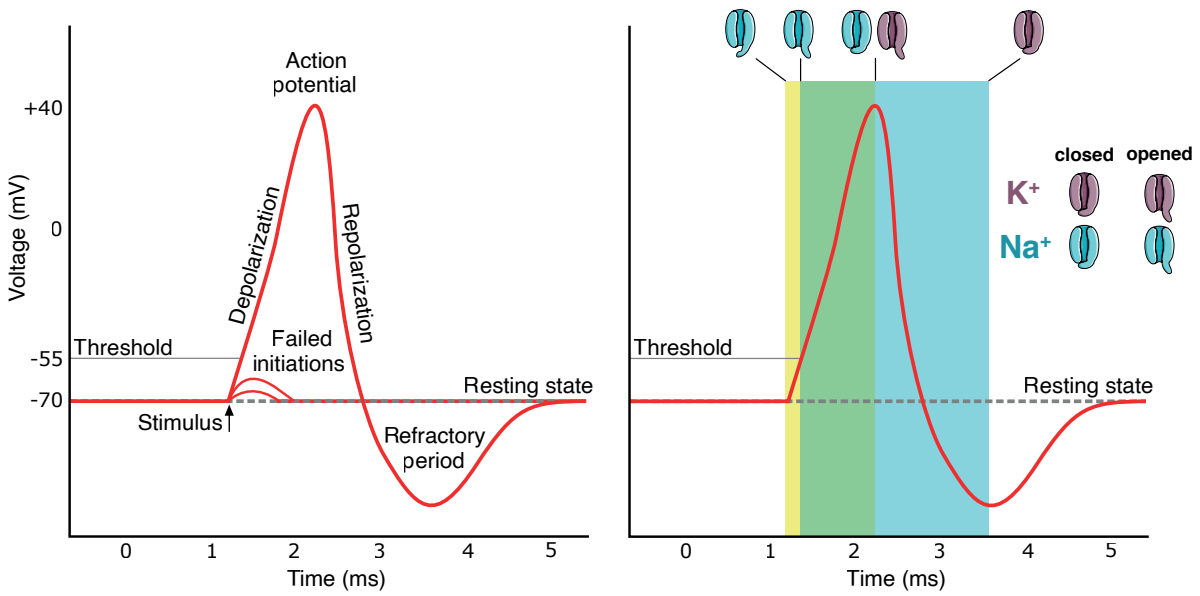


Figure 1.4: Different phases of the action potential and its correlation with the state of ion channels. The neuron emerges from the resting state upon reception of a stimulus that allows crossing of the threshold, causing depolarization through the opening of the Na^+ pump. The closing of Na^+ and the opening of the K^+ cause repolarization until it reaches hyperpolarization (refractory period) that ends with the closing of K^+ pump.

Depolarization. Upon stimulation, the membrane potential of the neuron will rise. If the stimulation stimulates the neuron enough for the membrane potential to cross the threshold of around -45 mV, the depolarization phase starts. As neurons work on an all-or-nothing logic, the neuron will return to its resting state if the threshold was not reached (see Figure 1.4). During depolarization, the Na^+ ion channels will open wider along with the rise of the potential, thus highly positively polarize the intracellular thanks to the entrance of Na^+ ions. The rise will stop before reaching its equilibrium potential of around 55 mV as voltage-dependent K^+ channels open.

Repolarization. The repolarization of the membrane is explained by the opening of the K^+ ion channels that repolarizes the membrane by exchanging ions with the extracellular. The membrane potential will then tend to the equilibrium potential of K^+ ($\sim$ -75 mV). The absolute refractory period is due to the inactivation of voltage-dependent Na+ channels. The relative refractory period is linked to excess hyperpolarization of the membrane potential after an action potential, as the membrane voltage tends towards the equilibrium potential of potassium ions.

Refractory period. Once the membrane potential passed below the resting potential the refractory period starts. The opening of the K^+ ion channels hyperpolarizes the membrane before spontaneously closing. Excess ions are evacuated, thus allowing the potential to slowly recover to the resting potential. The hyperpolarization of the membrane prevents the neuron from firing during the refractory period hence preventing backward propagation along the axon and limiting the firing frequency of the neuron.

1.2.8 Neural coding

Neural coding is a fundamental concept in neurosciences that refers to the processes of communication happening in brain based on action potentials. The nervous system encodes various

pattern from sensory inputs as well as motor output in patterns. The variability of neurons and the lack of repeatability of experiments have given rise to two major theories to explain the coding of patterns that either favors precise timing in the 'temporal coding' or the number of spikes, i.e. frequency in the 'rate coding'. The spatial distribution of the neuron activate in different regions is also considered to be part of the neural coding. Through neural coding, the brain transforms external stimuli into meaningful representations, enabling to process information and generate appropriate behavioral responses.

Temporal coding. The concept of temporal coding that associate a concept of precise timing to spikes so that the timing at which the action potential occurs is important. It is supported by works like [Abeles et al., 1993, Thorpe et al., 1996, Abeles and Gat, 2001]. These studies claimed that the behavior of neurons in the visual cortex rely on precise spike timings. While the variability of neurons and the lack of experiment repeatability question this concept based on the timing of action potentials, many phenomena are based on temporal coding.

Rate coding. The concept of rate coding that explains the neural coding in quantity of spikes emerged from [Adrian, 1926] that performed experiments on the nerve receptors of frogs. It was observed that the spiking frequency was increasing along with the pressure applied on the muscle, highlighting that the stimulus has an important effect on the frequency of the action potentials. [Brette, 2015a] describes the concept of "rate" in neural coding as an abstract mathematical construct of calculations over an infinite number of spikes for an averaged quantity defined by the timing of spikes.

These two theories raise the question of what is the most important characteristic of the action potential occurence in view of a communication with the living. Is the accurate reproduction of the biological shape of the action potential, guarantying a precise timing, the most important characteristic to capture? Or would a simpler model neglecting the shape of the action potential that relies on the frequency and average number of spikes be more appropriate?

Indeed, as previously mentioned, neurons are capable of generating an action potential from a stimulus as long as it enables the membrane potential to cross the threshold. Such a process would support the 'rate coding' that promote the frequency over the timing.

However, another line of thought on the same subject concerns the study of neurological diseases. Alterations of cells induced by neurological disorders in the nervous system affect the transmission of nerve impulses. This change not only affects the spiking frequency, it also affects the features of the action potential, thus supporting the consideration of the shape of the action potential as in 'temporal rate' theory.

1.3 Neurological disorders

Millions of people worldwide are affected by neurological disorders that strongly impair their cognitive and/or motor functions [Organization et al., 2020]. Globally, in 2019, there were nearly 10 million deaths and 349 million disability-adjusted life years (DALYs) due to neurological disorders [Ding et al., 2022]. They are for now untreatable and strongly linked with age, becoming a considerable challenge in the next years because of its significant cost in treatments that are limited to curbing the progress of the disease and managing the symptoms for a growing proportions of people affected. As the population is aging, the proportion of people affected by neurological disorders is expected to double in the next 20 years.

1.3.1 Neurodegenerative diseases

Neurodegenerative diseases or neurological diseases induce a progressive degeneration leading to the death/impairment of neuronal cells. They affect the central nervous system as well as peripheral nervous system. Neurodegenerative diseases can be caused by various factors, including brain lesions, tumors, infections or hereditary dysfunction. Some of the most common neurodegenerative diseases include Amyotrophic Lateral Sclerosis (ALS), Alzheimer's Disease (AD), Parkinson's Disease (PD) and Huntington's Disease (HD) [Checkoway et al., 2011]. Cognitive symptoms resulting from these diseases vary considerably but can also include problems with memory, speech, walking, coordination and vision. In order to better understand how models can help in the better understanding of the such diseases, the main diseases and their symptoms are briefly introduced below.

Amyotrophic Lateral Sclerosis. ALS leads to dysfunction of motor neurons in the spinal cord, cerebral cortex and brain stem. Symptoms include muscular weakness leading to paralysis, breathing inability and death. Among changes occurring during ALS progression, the morphology of motor neuron dendrites may be affected [Fogarty et al., 2016a], pointing out their role in the study of this disease. The initial impact of ALS affects the largest motor neurons (MN), specifically the fast-twitch fiber-controlling motor neurons (FF MNs).

Alzheimer's disease. AD causes loss of neurons and synapses and affecting glial cells and vascular system, leading to dysfunctions in the amygdala, hippocampus and other cortical areas. AD progressively affects cognitive abilities and memory. Synaptic plasticity as well as cortical neurons appears as essential elements to model in order to characterize the disease.

Parkinson's disease PD affects particularly dopamine-producing neurons in the brain and is thought to affect the striatum. These neurons are essential for controlling voluntary movement and coordination. Symptoms include motor symptoms such as tremors or postural instability as well as non-motor symptoms such as depression or sleep disturbances. The dopaminergic neurons and their interaction appears as essential to capture in a model to study the disease.

Huntington's disease. HD affects the projection neurons in the striatum, a nerve structure below the cortex. Neurons in this region have longer-than-average axons, up to 1 meter in length, that extend from the neuronal cell body within the central nervous system to one or more of its distant regions similarly to the spinal cord and cortex. Symptoms include the progressive onset of involuntary spontaneous movements and the gradual loss of cognitive abilities. HD has the particularity of affecting neurons with axons of a certain size, thus highlighting this parameter as essential in a model of study.

Other neurodegenerative diseases. Other disorders of the nervous system can be induced by abnormalities in specific ion channels morphology that exist in wide variety with specific functions and locations in the neuron [Toledo-Rodriguez et al., 2004].

[Lai and Jan, 2006] claims that any alteration in the ion channel morphology or location could affect communication in a neuronal network. Congenital anomalies affecting the ion channels, known as genetic channelopathies, could occur throughout the nervous system and create neurodegenerative disorders [Spillane et al., 2016]. The size of the neuron also constitutes a possible factor of nervous system disease.

[Kernell and Zwaagstra, 1981] demonstrates the influence of neuron size on the conduction, thus on the nerve impulses transmission speed in axons. Larger size of neuron is for example thought to be one of the symptoms of diseases such as Tuberous Sclerosis Complex (TSC) or Bourneville Tuberous Sclerosis [Ess, 2010] that lead to dysfunction at the level of synapses

[Bateup et al., 2013], neurons or larger size also are known to be the first affected in ALS.

All the neurodegenerative diseases presented highlight the importance of the morphology of neurons and its components in the study of neurodegenerative diseases. Therefore, consideration of the morphology of the neurons, dendrites, axons and ion channels is a mandatory criterion for an accurate model of neurodegenerative diseases.

1.3.2 Existing treatments

An increasing number of technologies and solutions are currently proposed for the treatments of neurodegenerative diseases, whereas being limited to curbing the progress or managing symptoms in most cases [Chin and Vora, 2014, French et al., 2016]. This is notably the case for pharmacological treatments that are still limited and show slow progress in the discovery of truly innovative molecules in the recent years [Rust et al., 2019].

As for an example of pharmacological treatments, in Alzheimer's disease, they have shown effectiveness in improving cognitive symptoms [Ballard et al., 2006] or managing moderate to severe Alzheimer's symptoms [Tariot et al., 2004].
[Gauthier et al., 2016] demonstrates that a selective inhibitor of tau protein aggregation shows efficiency in modifying disease progression in patients with mild to moderate Alzheimer's disease.
For ALS, the proposed treatment that helps to slow the disease progression is riluzole, which is intended to inhibit excess glutamate (excitotoxicity) and persistent Na+ current (INaP) [Miller et al., 2012]. For Huntington's disease it is shown effective in managing chorea [Frank, 2014]. However, some of these treatments may have a risk of potentially serious adverse effects as in [Frank, 2014]. More recently, a new anti-oxydative molecule named edaravone has been approved by the FDA and is used to treat ALS patients.

As for alternative treatments, deep brain stimulation (DBS) is for example used to manage Parkinson's symptoms [Deuschl et al., 2006]. Supportive care and therapies such as physical therapy and occupational therapy also play an important role in managing symptoms and enhancing quality of life for individuals affected by neurodegenerative diseases [Bennett et al., 2019, And, 2018].

Another approach considered is gene therapy that proposed various strategies to alter the expression of defective genes through DNA [Kolli et al., 2018, Cota-Coronado et al., 2019, Pahan, 2019].

1.3.3 Innovative alternative treatments

As the recovery of cognitive and motor functions of patients with disabilities is a global priority in healthcare and research [Semprini et al., 2018], the innovative treatments are favoring brain repair thus exploring brain rewiring to take better advantage of improved plasticity. In this context, new therapeutic approaches and technologies are needed both to promote cell survival and regeneration of local circuits [Farina et al., 2021] and restore long distance communication between disconnected brain regions and circuits [Bouton et al., 2016].

Recently, major progress has been made in the bioelectronics and neural engineering [Panuccio et al., 2018, Semprini et al., 2018] allowing the development of electroceutical-based devices [Famm et al., 2013, Reardon, 2014]. Neuromorphic devices such as neuroprostheses are now capable of receiving and processing input while locally or remotely delivering their output either

through electrical, chemical or optogenetic stimulation [Christensen et al., 2022]. Neuroprostheses are devices that allow direct interfacing of artificial circuits with large neuronal networks system to external technology, aiming to restore or enhance neurological functions in individuals with impairments by providing adaptive stimulation. Its bidirectional communication between artificial and biological is a promising feature that could make of it an interesting clinical solution for treating brain lesions [Broccard et al., 2017].

[Christensen et al., 2022] presents a roadmap on neuromorphic computing demonstrating that current and future neuromorphic systems will show capability of dynamic processing and learning of signals at low-power. Hence, it will pave the way to the integration of complex biological signal processing embedded in the new types of neuroprostheses.

Many neuromorphic interfaces, such as BMIs or BCIs, indeed exist. BMIs are systems that establish direct connection between the human brain and an external device such as a computer or a robotic system exist. System as presented in [Nicolelis and Lebedev, 2009, Hochberg et al., 2012, Bonifazi et al., 2013] concentrated on the interactions between neural chips enabling reproduction of similar electrical activities to neural networks and biological cells. The goal was to replace damaged areas of the brain by these systems by achieving acquisition and processing of brain signal and provide a response interpreted by the nervous system. These systems are said to be closed-loop, a crucial feature for research on neuroprostheses [Levi et al., 2018, Buccelli et al., 2019]. The Figure 1.5 illustrates the principle of biohybrid closed-loop systems.

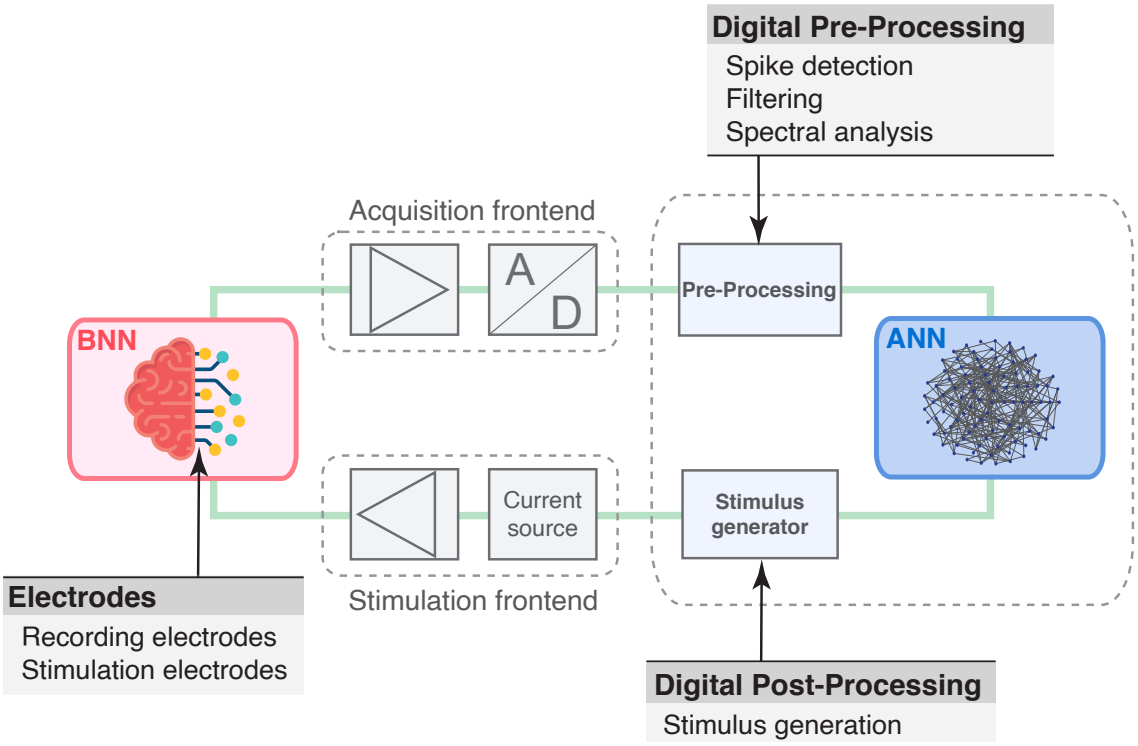


Figure 1.5: Example of biohybrid closed-loop system. Biological Neural Network (BNN) and Artificial Neural Network (ANN) are interfaced to allow bidirectional communication. The activity of the BNN is recording through electrodes later digitalized and processed to be fed to the ANN. A stimulation is generated from the activity of the ANN and sent as a current stimulation in the BNN through the stimulation electrodes. The activity of the BNN influence the activity of the ANN, this latter influencing back the BNN closing the loop.

1.4 Biological models

Modeling is a powerful tool for inquiry and discovery across numerous disciplines, it allows researchers to probe complex questions while avoiding constraints and outside influence. Biological models are experimental systems used by biologists to recreate specific biological processes.

1.4.1 Modeling human body

Biological models used to study the human body can be broadly categorized into in-vitro (cell-based) models and in-vivo (animal-based) models. It is crucial for our understanding of human body functioning and the pathologies that can affect it. As an example of animal-based model, Alzheimer's disease and ALS may be modeled using an animal model of transgenic mice [Elder et al., 2010, Julien and Kriz, 2006]. The main biological model categories are recapitulated in Figure 1.6.

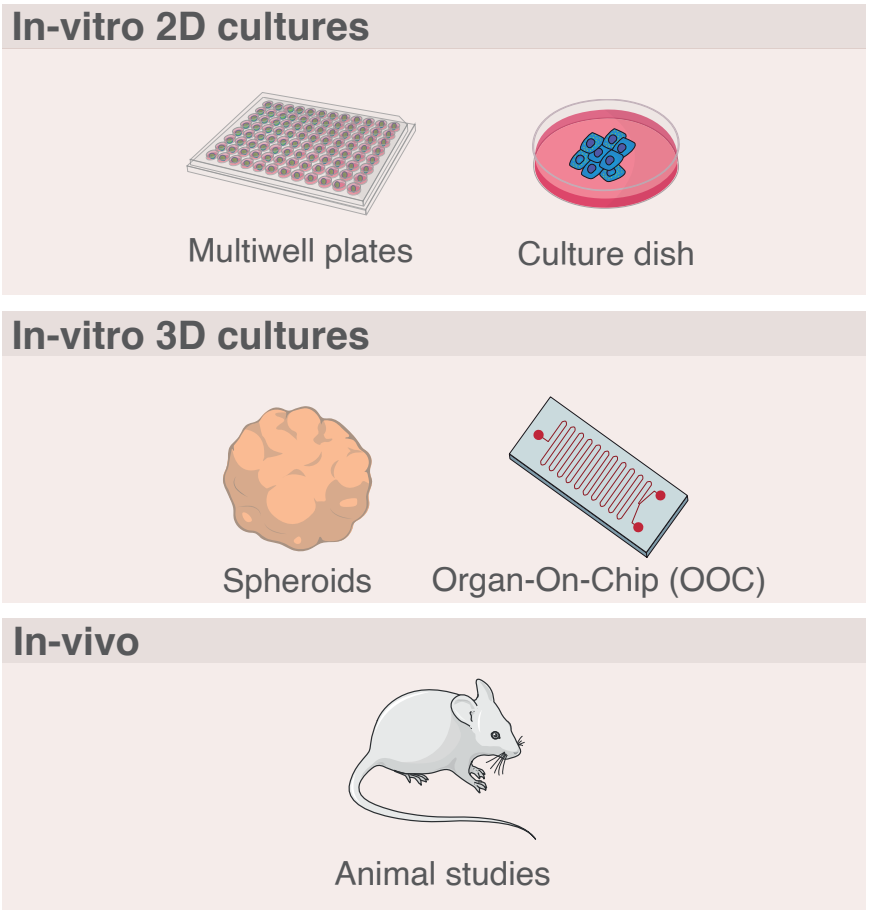


Figure 1.6: Main categories of biological modeling showing in-vitro culturing in two and three dimensions as well as in-vivo modeling through animal studies.

1.4.2 In-vitro 2D cultures

In-vitro two-dimensions cultures (2D) are cellular models widely used in biomedical research to study biological processes as well as human diseases. In-vitro 2D cultures are cultivated on flat surfaces such as culture dishes or multi-well plates thus allowing a consistent control of the environment.

The main benefit of 2D culturing lies in the simplicity of its handling and experimental measurements processes while providing a large range of established cell lines available. They notably allow the investigation of cellular interactions as well as drug screening. For instance, [Brewer et al., 2008] used a 2D culture of rat hippocampal neurons to investigate the mechanisms of synapse formation and maturation during early brain development. Another study utilized rat neuronal 2D cultures to investigate the underlying mechanisms of Amyloid β protein deposition, a hallmark of Alzheimer's disease [Lorenzo et al., 2000].

[Regnell et al., 2012] studied the accumulation of oxidative DNA damages as a potential cause of age-related cognitive decline by studying neural progenitor cells cultures of different animals.

[Kagan et al., 2022] proposed an experiment where 2D neuron cultures where used to study their learning ability when embodied in a simulated game-world.

However, this model is often limited in terms of physiological coherence as they do not fully reproduce the complexity of tissues and organs.

1.4.3 In-vitro 3D cultures

In-vitro three-dimensions cultures (3D) are cellular models that reproduce the structure and cell interactions in three-dimension observed in tissues and organs. In 3D cell culturing, cells can be cultivated in a device allowing the cells to organize and interact in a more coherent way thus providing a model of greater coherence [Pampaloni et al., 2007, Breslin and O'Driscoll, 2013, Duval et al., 2017]. It notably presents better representation of cell and tissue physiology, better cell differentiation, more realistic drug response and wider possibility of cell interactions. In-vitro 3D cultures are for example widely used to study human brain through organoids cultures that can reproduce structures of certain brain area [Kim et al., 2020].

A more complete model comes with design of Organ-On-Chip (OOC)s that refers to microfluidic platforms that aim to mimic the structure and function of human organs in vitro [Huh et al., 2011]. These devices incorporate cells and tissues in a three-dimensional arrangement to replicate the complex architecture and interactions present in real organs [Ma et al., 2021].

In organ-on-chip systems, cells are cultured in a 3D environment that can mimic the tissue-specific characteristics of organs such as the liver, lung or heart. The 3D cell cultures in OOC devices enables various interactions such as for cell-cell and cell-matrix interactions thus providing better reproduction of the microenvironment of the organ. They allow study at organ-level thus improving physiological coherence for processes like drug responses or disease modeling.

As for instance, [Choi et al., 2014] developed a three-dimensional (3D) cell culture model of Alzheimer's disease based on human neural progenitor cells (hNPCs), allowing to create a more physiologically relevant environment.

1.4.4 In-vivo experiments

In-vivo experiments refer to experiments or studies conducted on a living organism, usually an animal model or non-human organisms. A common animal model for human biology modeling being rodents [Peters et al., 2007]. In-vivo experiments also target the study of a variety of biological processes such as embryonic development, diseases modeling and drug responses. In-vivo experiments involve the study of biological processes and physiological responses in their actual physiological context, thus showing less limitations than in-vitro cultures.

The large benefit of in-vivo models lies in accounting for the complex interactions in a living organism. In-vivo cultures provide a more comprehensive approach to the understanding of biological processes and assessing the efficacy and safety of therapeutic interventions [Norrby, 2006].

Whereas in-vivo experiments show greater coherence and efficiency, they are often significantly more complex to conduct compared to in-vitro 2D cultures. Especially as in-vivo studies require rigorous ethical protocols to guarantee both the animal welfare and the scientific integrity of the research performed.

An example of work targeting Alzheimer's disease corresponds to transplantation of human PSCs cultivated in-vitro into the brain of a murine AD model. [Espuny-Camacho et al., 2017] taking advantage of 3D in-vitro cultures to control cell differentiation processes and in-vivo studies to retrieve the natural environment required for the study.

Other examples include [Nudo et al., 1996] that investigated plasticity of movement representations in the primary motor cortex of adult squirrel monkeys. [Garcia et al., 1995] used the rat middle cerebral artery occlusion model to simulate a stroke-like condition in rats by inducing a cerebral ischemia.

1.4.5 Data acquisition

Data acquisition and analysis of biological model varies greatly according due to the wide range of processes and models found in biology, especially when it comes to chemical or physical analysis. As this manuscript focuses on the neurodegenerative diseases and hybridization, therefore on a particular cell that is the neuron, only the relevant calcium- and voltage- imaging techniques will be presented.

The calcium- and voltage- techniques used to investigate neuronal activity can be divided in two categories: invasive or non-invasive. However, the invasiveness of techniques is directly linked with type of culture as for the in-vivo studies, physical constraints may prevent the use of certain non-invasive in-vitro techniques. Figure 1.7 recapitulates the techniques discussed in this section.

For in-vitro cultures, a widely used non-invasive technique relies on extracellular recording with voltage imaging through electrodes arrays ranging from tens of micrometers to the tens of nanometers depending on the technology.

The electrical activity captured by the electrodes is then sent to a recording unit that digitalizes the data to allow storage. The benefit of this method lies in its ease of use and low-level of invasiveness so as cultures can be cultivated for a long time on these devices.

Example of devices using this technology are MEA that record from electrode at the micrometer scale [Spira and Hai, 2020], NEA that shows greater resolution in signal-to-noise ratio [Larrieu and Han, 2013] and HD-MEA that allow greater spatial-resolution [Müller et al., 2015]. However, these techniques do not allow to directly capture action potentials.

Another technique is calcium imaging that uses microscopy technique to optically measure the calcium ($Ca2^{+}$) status that corresponds to spiking activity in the neurons [Grienberger and Konnerth, 2012]. As calcium imaging requires the use of a binding protein, it can be considered in a way more intrusive than extracellular recording. For in-vivo experiments, some non-invasive imaging techniques exist but often limited to certain cultures only so as invasive techniques is often used [Koo et al., 2006].

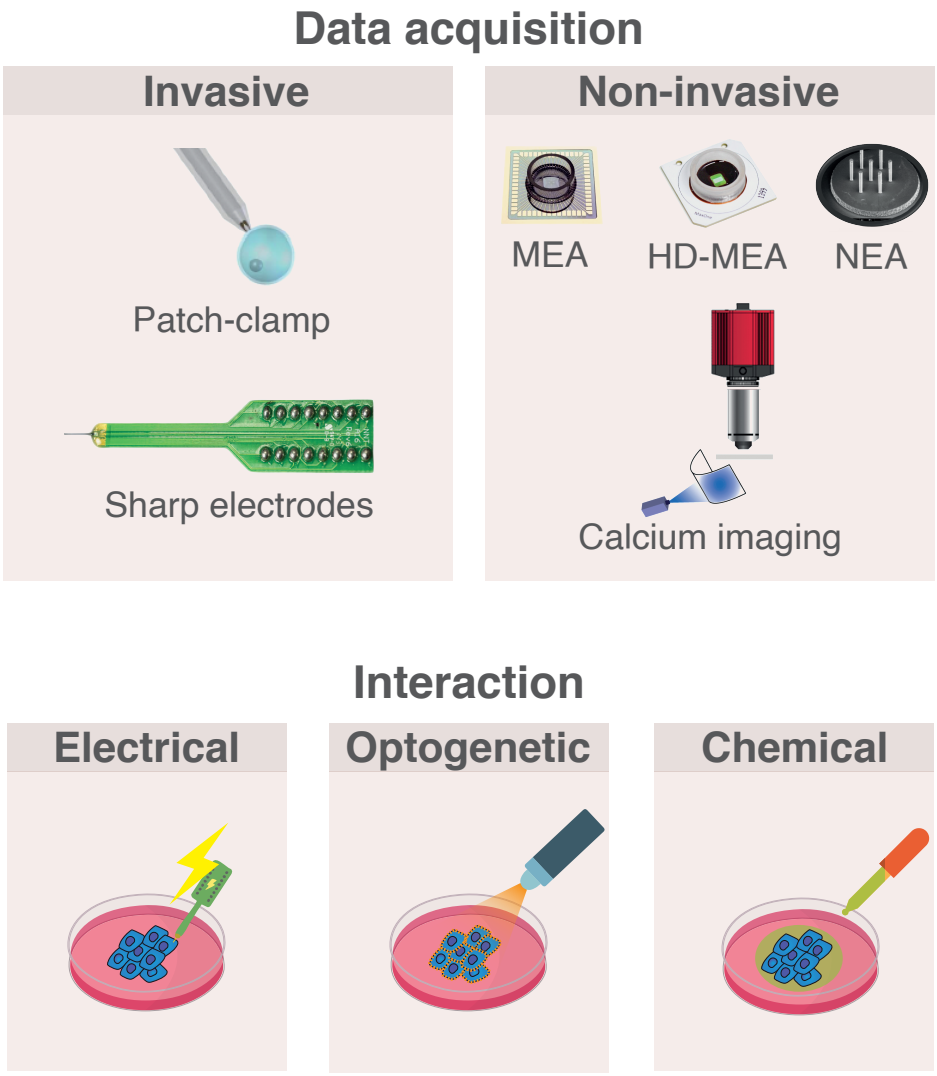


Figure 1.7: Main techniques of calcium and voltage imaging in-vitro to capture neuronal activity categorized by invasiveness and main interaction techniques. For in-vivo experiments, calcium imaging is invasive and electrode array-based techniques are indirectly used via the sharp electrodes.

While non-invasive techniques can provide coherent signal, invasive techniques that are closer to neurons provide signal of higher fidelity. In-vitro invasive methods include sharp electrodes that record electrical activity thanks to electrode arrays [Spira and Hai, 2020] and patch-clamp [Neher and Sakmann, 1992] that captures ionic currents. The sharp electrodes exist in numerous configuration for recording neural signals from different regions at the cellular level. The patch-clamp technique provides high resolution current recording of a cell thanks to a pipette and electrode, allowing direct recording of action potentials and ionic currents. However, these techniques are significantly more difficult to set up and cultures suffer from their invasiveness like neuron death in the case of patch clamp.

For in-vivo, patch clamp is doable [Furue et al., 2007] but yet difficult so as sharp electrodes are widespread. Because of physical constraints imposed by in-vivo experiments, the calcium imaging technique is invasively performed in-vivo [Gobel and Helmchen, 2007].

1.4.6 Model Interactions

A complex part of biological modeling lies in the interaction with the model. While the previous part was introducing methods and techniques to acquire data from the cultures, i.e. the output, a complex part of the model study corresponds to the choice of the input or stimulus. In the case of the biological models introduced in this section, three techniques are possible to interact with the culture: electrical, optogenetic and chemical (see Figure 1.7).

The electrical technique consists in an electrical stimulation applied to cultures often shaped as a biphasic pulse. It shows capability to activate neurons thanks to the current injected [Tehovnik, 1996]. This stimulation suits well sharp electrodes and electrode arrays as the electrode recording are often able to source stimulation. However, electrical stimulation may create artifact in recordings thus complicating the analysis of the behavioral response [Antal et al., 2014]. Paired with electrode arrays, it also lacks spatial accuracy as the stimulation spreads all over the culture.

The optogenetic technique corresponds to the introduction of stimulation using light on cultures genetically modified to response to specific wavelength. By introducing a light-gated membrane channel in neurons, it allows activation of the neuron using light [Nagel et al., 2003, Mosbacher et al., 2020]. The advantage of this stimulation lies in a better control of the cell stimulated through the availability of markers to observe the cells excitable.

Another technique to interact with the culture is using the chemical approach through drug treatments that affect the culture. Some drug treatments like bicuculline "disable" the inhibitor receptors of synapses leading to significantly higher activity that can lead to epilepsy [Ben-Ari et al., 1981]. Introduction of chemical may also be used as a way to generate micro-stimulation as in [Nishikawa et al., 2019].

The main challenges concerning the interaction with cultures lie in the assessment of its impact on the culture but mostly in its implementation that often require high cross-disciplinary skills and knowledge to obtain a satisfying setup in terms of coherency and noise robustness.

1.5 Artificial modeling

The consistent growth in computer performances that happened over the past years [Nordhaus, 2007] pushed the development and generalized the usage of computer simulated models also known as artificial models. An artificial model is a simplified representation created by human beings that aims to imitate or simulate a specific aspect of a system.

1.5.1 Bio-inspired and biomimetic approaches

Biomimicry or biomimetics consists in drawing inspiration from nature to solve human technological challenges. It is based on the study of properties and processes of the nature then adapted to create more performant technology. It is an approach widely used in various domain such as engineering, materials and energy. Nonetheless, biomimicry actually fuels two distinct approaches: the bio-inspired and the biomimetic approaches. While the bio-inspired approach rely on the inspiration from Nature to develop novel materials and devices, the biomimetic approach focuses on the mere reproduction of Nature and its replacement.

1.5.2 Neuron models

The complexity of physical, chemical and biological interactions of neurons led to numerous research works, a fair amount focusing on the interpretation and prediction of observations using so-called models. More specifically, the description of nerve impulses in neuroscience has been a source of numerous models varying in terms of their usefulness, complexity, level of detail and the behavior described.

Neuron modeling is a good example of the bio-inspired and biomimetic approaches. Bio-inspired neuron models inspired by biology enabling calculation such as image recognition, classification or data processing differ from biomimetic neurons that aim to reproduce faithfully phenomena happening in nature.

The first biologically meaningful mathematical neuron model was proposed by [Hodgkin and Huxley, 1990] known as HH model. It was designed from experiments performed on the squid giant axon that involved three currents: sodium, potassium and leak current. It was demonstrated that the ionic permeability of the membrane was highly dependent on the membrane potential, thus leading to a mathematical model based on the voltage-dependent properties of the membrane to generate an action potential. The opening and closing of the ion channels are then translated as a variable conductance depending on the voltage. The schematic diagram of the Hodgkin-Huxley model is shown in Figure 1.8 where V_m is the membrane potential, C_m is the membrane capacitance, g_{Na} and g_K the conductance of voltage-dependent ion channels, E_{Na}, E_K and E_{Leak} are the reversal potentials and g_{Leak} a voltage independent conductance.

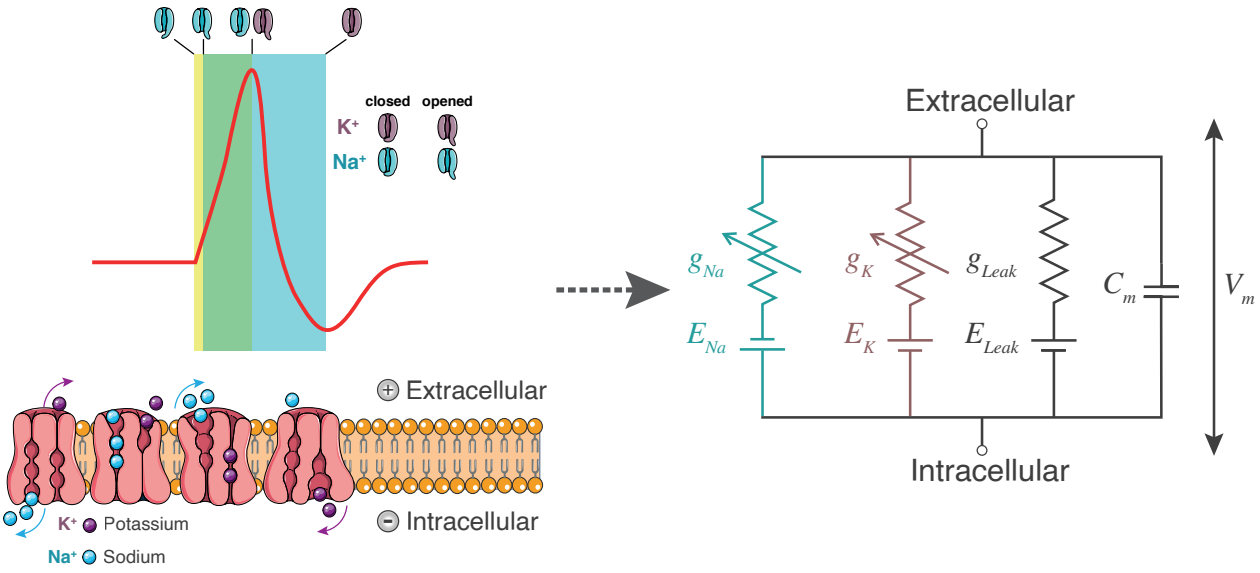


Figure 1.8: Electrical equivalent circuit of HH model representing Na^+, K^+ ionic currents and leakage current as a basic neuron model.

Hence, from the schematic shown in Figure 1.8, the evolution of the membrane potential is ruled by current flowing across the membrane integrated by the membrane capacitance according to Equation 1.1:

$$C_m \frac{dV_m}{dt} = -(I_{Na} + I_K + I_{Leak}) \tag{1.1}$$

where, I_{Na} is the sodium current, I_K the potassium current, I_{Leak} the leakage current. The

HH model will be further detailed in Section 1.5.4

As the HH model is based on high-dimensional nonlinear differential equations, it is considered as difficult to analyze with commons studies such as phase plane analysis, various model simplified models of HH were introduced. On the contrary, other widely used models of spiking neurons are based rather on a threshold and reset mechanism to produce action potentials.

Examples of simplified derived models from HH model. The FitzHugh-Nagumo model [FitzHugh, 1955, FitzHugh, 1961, Nagumo et al., 1962] simplified the gating variables of slow kinetics based on observations while conserving many qualitative characteristics.

The Morris-Lecar model [Morris and Lecar, 1981] that combine both HH model and FitzHugh-Nugamo model into a voltage-gated calcium channel model with a delayed-rectifier potassium channel, thus describing complex relationship between membrane potential and the activation of ion channels within the membrane.

Examples of other widely used models of spiking neurons. The simplest bio-inspired model is the IF model [Abbott, 1999] that produces a spike when the membrane potential crosses the threshold then reset to the resting potential.

The LIF model [Gerstner and Kistler, 2002] introduces a leak term to the previous model, thus allowing time-dependent memory of the stimulus where the IF keeps stimulus forever until spiking.

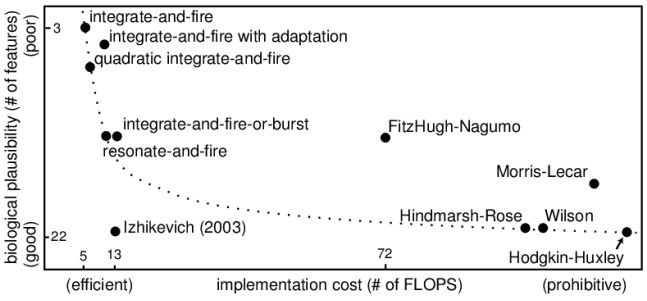
These models can also be enriched to add more biological plausibility in EIF [Fourcaud-Trocmé et al., 2003], Adex [Gerstner and Brette, 2009] models or quartic model [Touboul, 2008] that add terms allowing better reproduction of shape of the action potential.

The model allowing a significantly better biological plausibility is the IZ model [Izhikevich, 2003] that allows reproduction of various neuronal activities as shown in Figure 1.11 based on the principle of threshold and reset mechanism paired with a membrane recovery variable.

While each one of these neuron models translates to a certain level of biological coherence thanks to the number of possible biological phenomena reproducible as classified by the Figure 1.9 from the work [Izhikevich, 2004], only few can be considered as truly biomimetic. Considering the biophysical meaningfulness as an essential criterion for true biomimetic modeling [Brette, 2015b], the most biomimetic model is the conductance-based model HH [Hodgkin and Huxley, 1990].

However, for an implementation on a numeric platform and as demonstrated in Figure 1.9, the dilemma between biological coherence and implementation rises. Most certainly, the IZ model presents a very appealing compromise between implementation cost and biological coherence. Nonetheless, it lacks biological meaningfulness and the work [Brette, 2015b] claims that the model moves away from bio-realism and its ability to predict the behavior of neurons in certain conditions which contradicts the articles of Izhikevich.

As the manuscript focuses on biomimetic neuron models in view of biologically coherent emulation of neural network in the context of neurological disorders studies, biological meaningfulness is an essential criterion that strongly suggest the use of the HH model. Indeed, the shape of the action potential in such model allows a high coherence with biology emphasized by its biological meaningfulness.



Models	biophysically meaningful	tonic spiking	phasic spiking	tonic bursting	phasic bursting	mixed mode	spike frequency adaptation	class 1 excitable	class 2 excitable	spike latency	subthreshold oscillations	resonator	integrator	rebound spike	rebound burst	threshold variability	bistability	DAP	accommodation	inhibition-indiced spiking	inhibition-induced bursting	chaos	# of FLOPS
integrate-and-fire	-	+	-	-	-	-	-	+	-	-	-	-	+	-	-	-	-	-	-	-	-	-	5
integrate-and-fire with adapt.	-	+	-	-	-	-	+	+	-	-	-	-	+	-	-	-	-	+	-	-	-	-	10
integrate-and-fire-or-burst	-	+	+		+	-	+	+	-	-	-	-	+	+	+	-	+	+	-	-	-		13
resonate-and-fire	-	+	+	-	-	-	-	+	+	-	+	+	+	+	-	-	+	+	+	-	-	+	10
quadratic integrate-and-fire	-	+	-	-	-	-	-	+	-	+	-	-	+	-	-	+	+	-	-	-	-	-	7
Izhikevich (2003)	-	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	13
FitzHugh-Nagumo	-	+	+	-		-	-	+	-	+	+	+	-	+	-	+	+	-	+	+	-	-	72
Hindmarsh-Rose	-	+	+	+			+	+	+	+	+	+	+	+	+	+	+	+	+	+		+	120
Morris-Lecar	+	+	+	-		-	-	+	+	+	+	+	+	+		+	+	-	+	+	-	-	600
Wilson	-	+	+	+			+	+	+	+	+	+	+	+	+		+	+	+				180
Hodgkin-Huxley	+	+	+	+			+	+	+	+	+	+	+	+	+	+	+	+	+			+	1200

Figure 1.9: Comparison of the neuro-computational properties of spiking and bursting models neuron model from [Izhikevich, 2004]. "# of FLOPS" is an approximate number of floating point operations (addition, multiplication, etc.) needed to simulate the model during a 1 ms time span. Each empty square indicates the property that the model should exhibit in principle (in theory) if the parameters are chosen appropriately, but the author failed to find the parameters within a reasonable period of time.

1.5.3 Spontaneous activity

A fundamental property of nervous systems that is important to reproduce is the spontaneous activity of neurons that refers to the firing of neurons in the absence of sensory input. A common approach to describe this non-evoked or stimulus-independent activity is to use randomized laws to enable spontaneous activity through random spiking.

This random behavior can be modeled by injecting a noisy current into the neural network, usually based on a normal distribution. This noisy current will trigger spikes more or less frequently and on a random basis depending on its parameters like strength and deviation. A more biologically coherent noise than normally distributed can be generated by fluctuating the conductances of ion channels [Destexhe et al., 2001, Tuckwell et al., 2002], creating a current similar to synaptic noise observed in biology. Thus, a biomimetic noise based on Ornstein-Uhlenbeck process correspond to Equation 1.2.

$$\frac{di_{noise}(t)}{dt} = \theta(\mu - i_{noise}(t)) + \sigma \frac{dW(t)}{dt} \tag{1.2}$$

where, i_{noise} is the noisy current, θ and σ are parameters of the noise tuning its amplitude and deviation, μ is a constant adding drift and W_t denote the Wiener process.

1.5.4 Single compartment modeling

The most widespread representation of neurons in computing science is based on the representation of the neuron as a point in space, thus focusing on its temporal dimension. A common approach is to approximate the neuron as cylinder for which the membrane voltage is computer at the middle. The single compartment modeling of a neuron modeled using the HH paradigm [Hodgkin and Huxley, 1990] can be equated as shown in Equation 1.1 to describe a simple FS. This basic neuron translates the essential dynamics of the Na^+ and K^+ channels.

The ionic currents are mimicked by using variable conductance representing the opening state of the ion channels and voltage generator to represent the equilibrium potential of the ions (see Figure 1.8).

$$I_{ion} = g_{ion} m_{ion}^n h_{ion}^k (v - E_{ion}) \tag{1.3}$$

where, I_{ion} is the ionic current, g_{ion} the maximum conductance of the ion channel, m_{ion} and h_{ion} the probabilities between 0 and 1 respectively of ion channel activation and inactivation, v the membrane voltage and E_{ion} the equilibrium potential of the ion.

More specifically, the ionic currents allowing the reproduction of a FS neuron corresponds to the Equations 1.4,1.5,1.6.

$$I_{Na} = g_{Na} m_{Na}^3 h_{Na} (v - E_{Na}) \tag{1.4}$$

$$I_K = g_K m_K^4 (v - E_K) \tag{1.5}$$

$$I_{Leak} = g_{Leak} (v - E_{Leak}) \tag{1.6}$$

The equations ruling the probabilities of the activation and inactivation of the voltage-gated ion channels take the form a sigmoid with different dynamics for each ion. The equations are often expressed using the two formalism shown of Equation 1.7 or Equation 1.8.

$$\frac{dx}{dt} = \alpha_x(V)(1 - x) - \beta_x(V)x \tag{1.7}$$

$$\frac{dx}{dt} = \frac{x_\infty(V) - x}{\tau_x(V)} \tag{1.8}$$

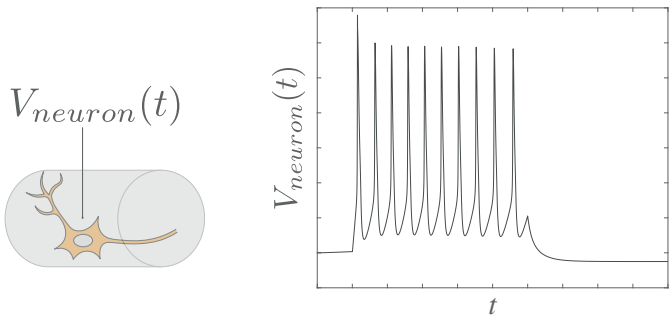


Figure 1.10: Single compartment modeling of a neuron. The neuron is represented as a cylinder in which membrane voltage is computed at one point in space. The membrane voltage V_{neuron} is only depending on time t as shown on the graph representing spiking activity.

The dynamics of the neuron can be enriched from this basis to mimic other neuron families as shown in Figure 1.11. For example, adding slow potassium current [Yamada et al., 1989] creates the RS neuron. Incrementally adding a slow calcium current [Reuveni et al., 1993] creates the IB neuron, while adding low-threshold calcium [Jahnsen and Llinás, 1984, Pospischil et al., 2008] creates the LTS neuron. This highlights the ability of the HH neuron model to link the action potential shape to accurate biological meaningfulness.

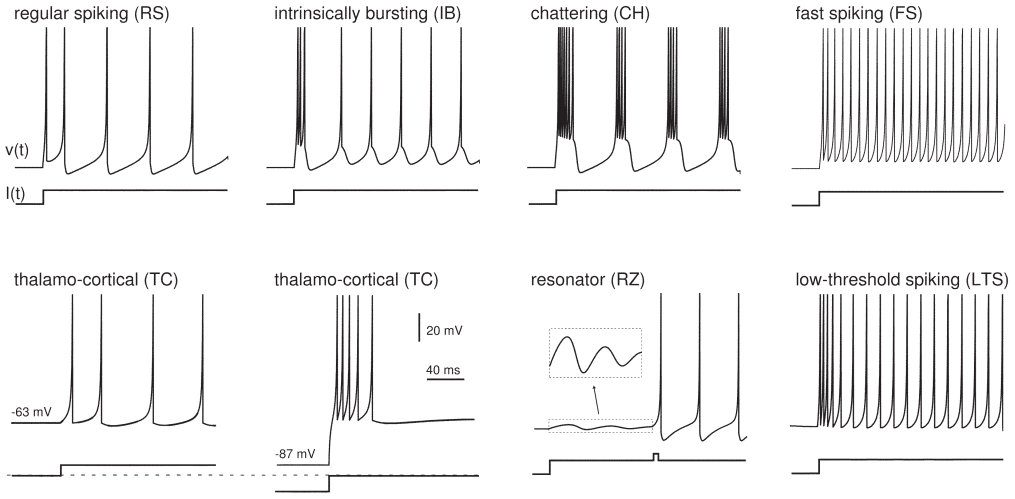


Figure 1.11: Simple model of spiking neuron from [Izhikevich, 2003]. Each neuron family is characterized by the evolution of its membrane voltage to a stimulation step.

1.5.5 Multicompartmental modeling

Single compartment models are the most widely used, as they are less resource-intensive and relatively efficient in most cases. Single compartment modeling, while demonstrating high prediction rates of action potentials and biophysical coherence especially in HH model, remains limited due to its inability to capture the complex spatial details and behaviors of neurons.

In contrast, multicompartmental models offer a more comprehensive and biologically realistic approach, thus providing deeper insights into neuronal function and information processing (see Figure 1.12).

It is particularly important as regions such as dendrites are the center of vital computations linked to their spatial morphology [Forrest et al., 2018] and are affected by some neurodegenerative diseases like ALS [Fogarty et al., 2016a].

Moreover, studies like [Brette, 2015b] shows that there are phenomena such as frequency-dependent attenuation of membrane as a function of frequency or the presence of wide variations in voltage which may be induced by the presence of active conductances distributed along the axon and dendrites.

Thus, important biophysical phenomena like spike initiation [Naundorf et al., 2006] in the AIS [Debanne et al., 2011] or in the dendrites [Gasparini et al., 2004] can be modeled. Phenomena like dendritic spikes are for example known to play a part in stimulus selectivity in cortical neurons [Smith et al., 2013] shows the importance.

Multicompartmental modeling also allows the investigation of the role of dendrites in neurons, the role of dendrites being a source of much research. [Mel, 1999] suggests that dendrites exist to increase the surface area of neurons increasing the possible number of synaptic connections through extensions of 10 to 20 times the surface area of the soma. [Chklovskii et al., 2002]

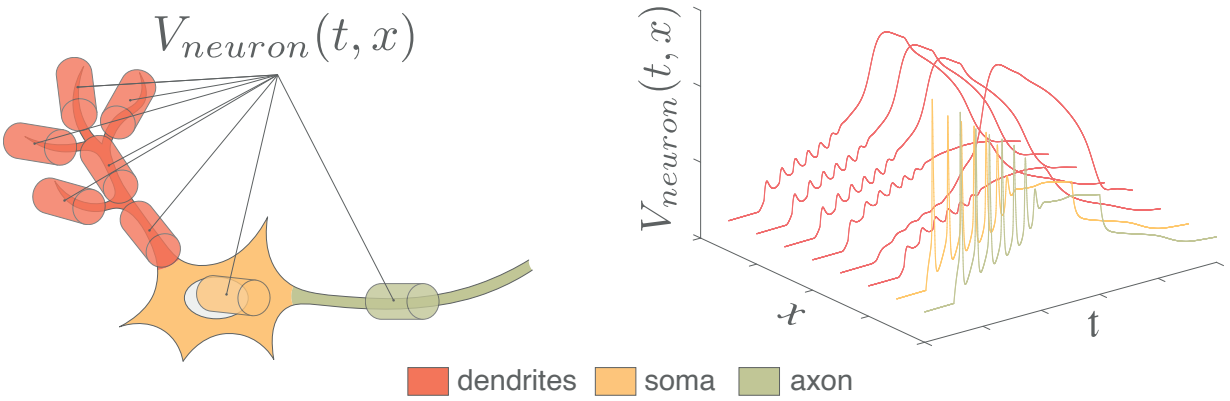


Figure 1.12: Representation of multicompartmental modeling that shows the different parts of the neuron modeled as connected cylinders and the difference in the signal waveforms depending on its location.

shows that the circuit connections in the cortical region are optimal and that the proportions are all more or less the same. Other studies like [McBain and Fisahn, 2001] support the hypothesis that dendrites exist in order to have several possible and separate entries on the surface of nerve cells. Dendrites also allow a greater diversity of presynaptic terminal classes creating different learning laws [Froemke et al., 2005]. They are also known to display physiological and morphological abnormalities during postnatal development in motor neurons with ALS [Martin et al., 2013].

The multicompartmental modeling applied to HH model is based on the one dimensional cable equation and corresponds to Equation 1.9, thus introduction spatial dimension x in the equation.

$$\frac{1}{2\pi a}\frac{\partial}{\partial x}\left(\frac{\pi a^2}{R_a}\frac{\partial V}{\partial x}\right) = C_m\frac{\partial V}{\partial t} + I_{HH} \tag{1.9}$$

where, a is the radius of the compartment, R_a the resistance of the axon, C_m the membrane capacitance, I_{HH} the currents of the HH model and V the membrane potential in the middle of the compartment.

A common approach for discretization is "compartmentalization" that approximates the cable equations by a series of compartments connected by resistors (see Figure 1.12,1.13) [Carnevale and Hines, 2006]. An electrical equivalent circuit of this multicompartmental model using HH paradigm is shown in Figure 1.13. Using this approach, membrane voltage is evaluated at the middle of each compartment.The discretized model could also be seen as the computation of spatio-temporally continuous variables over a set of discrete points in space ("grid" of "nodes") for a finite number of instants in time [Carnevale and Hines, 2006].

As compartments now correspond to different elements of the neuron, their properties can be translated by the HH model as illustrated on Figure 1.13. The soma if for example implementing 3 ionic currents with certain properties where a dendrite for example implement only one ionic current with different properties. This is a crucial feature that allow to model changes affecting only certain elements of the neuron like dendrites on specific channels through corresponding parameters tuning. The spatial morphology of the dendrites and axons can also be reproduced by the length of the compartment. This characteristic is notably important in the impact on the morphology of dendrites in ALS [Fogarty et al., 2016b] or neurodegenerative diseases like

Huntington's disease that affect neurons with long axons.

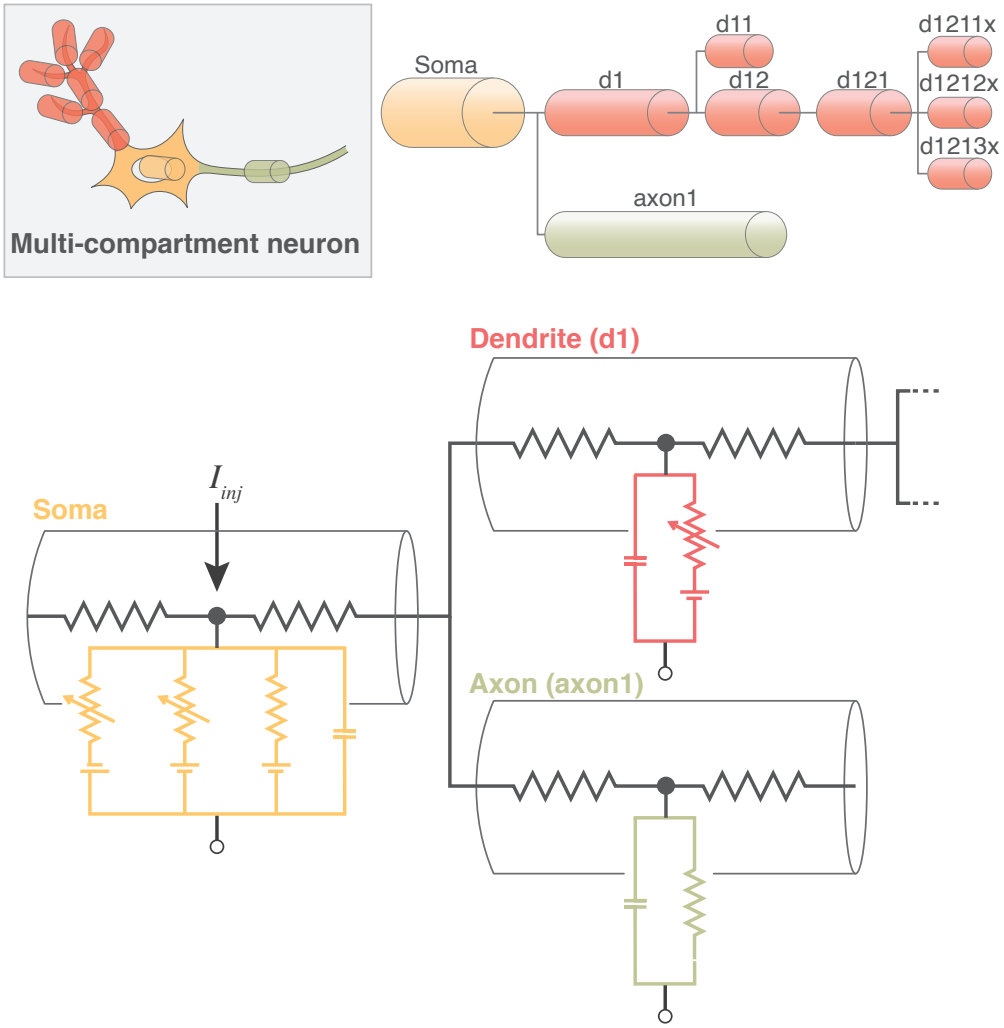


Figure 1.13: Electrical equivalent circuit of multicompartmental neuron model. The neuron is compartmentalized in cylinder of various length and diameters representing different elements of the neuron and their properties. I_{inj} is the current injected.

The multicompartmental modeling is undoubtedly crucial to the creation of faithful and reliable model providing enough biological meaningfulness to study neurological disorders through artificial models.

1.5.6 Synapse models

The modeling of nerve impulses in neurons is not limited to neuron models, numerous synapse models that address the same problematics as neuron models are found in literature. A wide variety of synapse models with different level of biophysical coherence exists that translate different dynamics of the synapse from the excitation and inhibition to plasticity.

[Izhikevich, 2003] presents a noise-injected synapse to reproducing the stochastic behavior of biological synapses, while [Cassidy et al., 2011] presents a synapse based on exponential current. [Rice et al., 2009] presents a synapse model that enables learning using long-term plasticity or short-term plasticity [Izhikevich and Edelman, 2008].

In the biophysically detailed models, a biomimetic and biologically meaningful model corresponds to [Destexhe et al., 1998], where four types of synaptic receptors were modeled: AMPAR, NMDAR, $GABA_AR$ and $GABA_BR$. They correspond to synaptic receptors found in dendrites involved in chemical synapses that are responsible for fast and slow excitation (AMPAR and NMDAR) and fast and slow inhibition ($GABA_AR$ and $GABA_BR$). The model translates the opening and closing states of the receptors based on a conductance-based model that provides biological meaningfulness (see Figure 1.14).

In this manuscript, the synapse model selected is [Destexhe et al., 1998] for its biological coherence and meaningfulness. As it is a conductance-based model, the equations of the currents are similar to the HH neuron model. Equations 1.10,1.11,1.12,1.13 correspond to the synaptic current generator by the receptors.

$$I_{AMPA} = g_{AMPA} \times r \times (V - E_{AMPA}) \tag{1.10}$$

$$I_{NDMA} = g_{NMDA} \times B(V) \times r \times (V - E_{NDMA}) \tag{1.11}$$

$$I_{GABAa} = g_{GABAa} \times r \times (V - E_{GABAa}) \tag{1.12}$$

$$I_{GABAb} = g_{GABAb} \times \frac{s^n}{s^n + K_d} \times (V - E_{GABAb}) \tag{1.13}$$

where, g_{AMPA}, g_{NMDA}, g_{GABAa}, g_{GABAb} are the maximum conductances of the receptors, E_{AMPA}, E_{NMDA}, E_{GABAa}, E_{GABAb} the equilibrium potentials, r and s the states variables for activation and inactivation of the receptors.

An essential property for synapses is its learning ability explained by synaptic plasticity. While synaptic plasticity may be ruled by several laws [Bono and Clopath, 2017], the synaptic rules widely used are STP and STDP.

STP refers to the dynamic changes in synaptic strength that occur over short periods, typically ranging from milliseconds to seconds. It involves mechanisms like facilitation and depression which impact the efficacy of synaptic transmission. STP plays a crucial role in regulating the temporal dynamics of neural information processing. A biomimetic and biologically meaningful model of STP based on the synapses involved in CPG network is presented in [Hill et al., 2001] using a conductance-based model.

On the other hand, STDP is a learning rule based on spike timing between neurons rather than a timing window. It correlates presynaptic spiking with postsynaptic neuron spiking, so as the synapse strength increase if the presynaptic spike consistently precedes the postsynaptic spike [Song et al., 2000]. The opposite situation leads the synapse strength to weaken thus decreasing its efficiency. STDP is rule widely used in artificial neural networks to learn temporal patterns and associations that suits well tasks involving temporal data such as speech recognition or sequence learning [Kheradpisheh et al., 2018].

Another concept that is the Hebbian plasticity relying on the rule "cells that fire together, wire together" increase the synapse strength when two neurons are activated simultaneously or in close temporal proximity [Abbott and Nelson, 2000]. While it is considered biologically plausible as it takes into account the activity patterns of neurons, it is considered to be a simplistic rule that does not account for certain aspects of learning, such as specificity and stability.

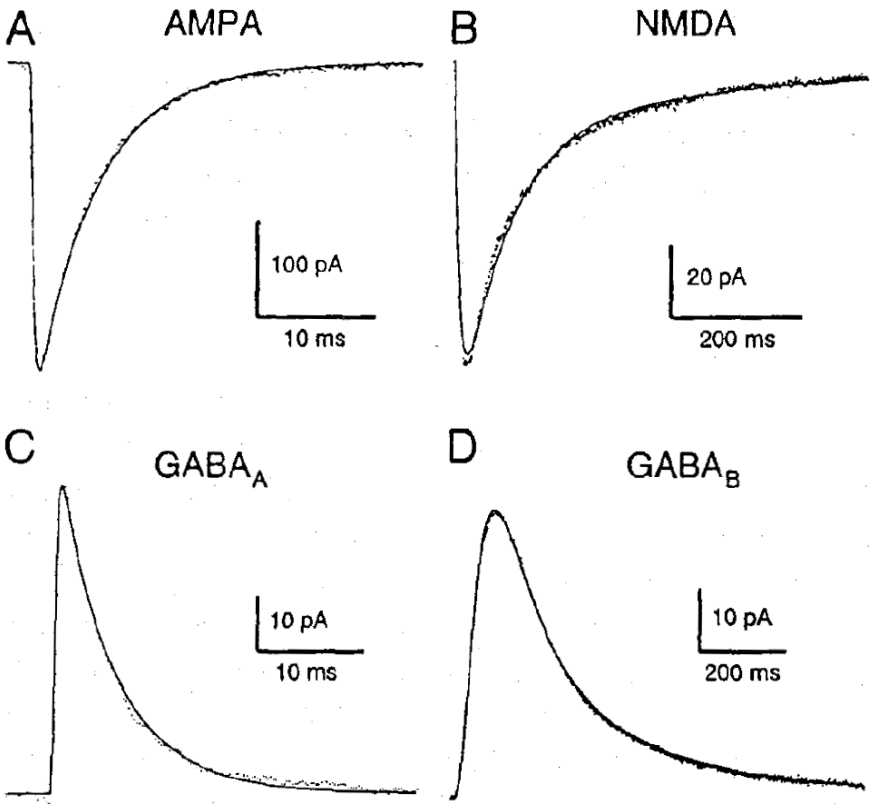


Figure 1.14: Post-synaptic currents graph from [Destexhe et al., 1998]. The curves represent the best fits of detailed kinetic models to averaged biological postsynaptic current signals obtained from patch clamp recording. (**A**) AMPAR current (**B**) NMDA current (**C**) $GABA_AR$ (**D**) $GABA_BR$.

In order to create a truly biologically coherent neural network, multiple synapses models should be used to reproduce biology. The implementation of both STP and STDP would allow mimicking learning behavior. A synapse model with high biological coherence like Destexhe is also important to reproduce the various dynamics based on the ratio and speed of inhibition and excitation in the network as well as emulating interventions such as drug treatments targeting specific synaptic receptors. Hence, the organization of neurons into a network also introduces the notion of biological coherence through the synapse model used to allow reproduction of various mechanism of the biology (plasticity, synaptic transmission, inhibition and excitation ratio, ...).

1.5.7 Neural networks

Neural networks is a broad definition referring to either a neural circuit of biological or artificial neurons. As for ANNs, numerous architectures serving different applications exist.

Bio-inspired ANNs that draw inspiration from human brain architecture to perform signal processing tasks are numerous. Tasks such as image recognition and classification or facial recognition often involve Feedforward Neural Network (FNN) [Svozil et al., 1997] or Convolutional Neural Network (CNN) [O'Shea and Nash, 2015] that rely on information processing by layers. As for instance, [Kheradpisheh et al., 2018] demonstrates the use of STDP-based CNN for object recognition. Other tasks like natural language processing, speech recognition, and time-series prediction tasks often involve Recurrent Neural Network (RNN) [Medsker and Jain, 2001].

In the biomimetic ANNs, SNN attempt to mimic the communication between neurons using action potentials or spikes [Ghosh-Dastidar and Adeli, 2009], thus being the closest to the behavior of biological neurons and their interactions. Liquid State Machines (LSMs) [Maass et al., 2002] are a type of RNN inspired by the liquid state computation to replicate its computation with randomly connected neurons.

As research in this field progresses, biomimetic neural networks hold promise for enhancing the performance and efficiency of artificial intelligence systems improving performances in applications across various domains.

1.6 Numerical system solving for artificial neuron modeling

A crucial step in the implementation of artificial neuron models is the solving of the differential equations ruling the evolution of the membrane potential using solver. As digital computing is inherently discrete, the use of numerical solver to find numerical solutions to the equations of neurons that are continuous in time and space are required. Therefore, implementing a model of neuron on a digital platform raises many purely numerical issues not related to biological questions but of crucial importance to the coherence of the simulations. Different types of solver exist, each with its own advantages and limitations involving the notions of complexity, accuracy and stability.

1.6.1 Numeric solvers

Numerical solvers are computer algorithms that find approximate solutions to complex mathematical equations. They are used to solve problems such as differential equations, algebraic equations and optimizations. Numerical solvers can be based on analytical methods such as the Taylor series method or on numerical methods such as the finite difference method. Numerical methods are often used when equations cannot be solved analytically or when accurate and reliable solutions are required. Contrary to analog solving that is highly resource-intensive on a digital platform, numeric solving suits better the architecture of digital platforms thanks to its discrete property. The solvers can be categorized in two categorized being the explicit and implicit solvers.

Explicit solvers. Explicit solvers are numerical algorithms relying on numerical schemes such as the Forward Euler method or the Runge-Kutta method to find a numerical approximation to the exact solution. They are very efficient when used to solve short-time differential equations but can be unstable for long time differential equations due to the strong dependence on the time step size [Courant et al., 1967]. The smaller the time step compared to the dynamic of the system, the better the stability. Nonetheless, a small time step often correlates with higher implementation cost.

Implicit solvers. Unlike explicit solvers, implicit solvers require knowledge of the exact solution at a given time in order to find the solution at a later time. Implicit solvers rely on numerical schemes such as the Backward Euler method or the Crank-Nicolson method to find a numerical approximation to the exact solution. They are more stable than explicit solvers for long-term differential equations but are often more expensive in terms of computation time.

1.6.2 Single compartment modeling

In the case of the single compartment modeling, the evolution of the membrane potential is ruled by a single ordinary differential equation (see Equation 1.1). Thus, only the discretization with

respect to time is required. The numerical integration methods the most used in empirically-based neuron modeling that will be discussed in this manuscript are Forward Euler, Backward Euler and Crank-Nicholson [Carnevale and Hines, 2006]. These methods are based on the finite difference expression as equated in Equation 1.14. However, other methods like CVODE or Runge-Kutta variants are also used.

$$\frac{dV}{dt} \approx \frac{V(t+\Delta t) - V(t)}{\Delta t} \tag{1.14}$$

Forward Euler. The Forward Euler is an explicit method characterized by [Carnevale and Hines, 2006] as the simple, inaccurate and unstable method. It is based on a simple approximation that starts from something already known and projected into the future, thus calculating the future value entirely on the basis of past and present. This method approximates a solution by applying the Equation 1.15, that applied to the HH model can be equated as in Equation 1.16. Forward Euler has first order accuracy that means that the local error is proportional to Δt and can allow stable simulation of single compartment modeling if its value is small enough [Khoyratee et al., 2019].

$$V(t+\Delta t) = V(t) + f(V(t), t)\Delta t \tag{1.15}$$

$$V^{n+1} = V^n - \frac{\Delta t}{C_m} \sum I_{ion}(t) \tag{1.16}$$

Backward Euler. The Backward Euler is an implicit method characterized by [Carnevale and Hines, 2006] as the inaccurate but stable method. As it is an implicit method, it involves the futures values in the calculation as shown in Equation 1.17 that involves $f(V(t+\Delta t), t+\Delta t)$. The great advantage of this method lies in its robust stability that prevent oscillations of the solution.

$$V(t+\Delta t) = V(t) + f(V(t+\Delta t), t+\Delta t)\Delta t \tag{1.17}$$

Crank-Nicholson. The Crank-Nicholson is an implicit method combining both the Forward and Backward Euler and characterized by [Carnevale and Hines, 2006] as stable and more accurate method. The expression of the Crank-Nicholson is shown in Equation 1.18.

$$V(t+\Delta t) = 2V(t+\frac{\Delta t}{2}) - V(t) \tag{1.18}$$

Among the three solvers, the most suitable for embedded platforms of single compartment neurons that features correct accuracy for a low implementation cost is the Forward Euler because of its simple explicit solving. It is also known to show satisfying stability for the emulation of single compartment neurons with a small time step [Khoyratee et al., 2019].

1.6.3 Multicompartmental modeling

Similarly to single compartment modeling, continuous solving is highly resource-intensive on a digital platform so as both spatial and temporal discretization are often required, especially as the equation now involves two independent variables. The most common spatial discretization involves the second order correct approximation of $\partial^2 V/\partial x^2$ (see Equation 1.9) shown in Equation 1.19. A representation is illustrated in Figure 1.15 where a cable also called section is discretized in compartments.

$$\frac{\partial^2 V}{\partial x^2} \approx \frac{V(x+\Delta x) - 2V(x) + V(x-\Delta x)}{\Delta x^2} \tag{1.19}$$

The discretized model can be seen as the computation of spatio-temporally continuous variables over a set of discrete points in space ("grid" of "nodes") for a finite number of instants in time [Carnevale and Hines, 2006]. Therefore, values of functions will refer at points on the grid function corresponding to the expression 1.20

$$G_i^n \equiv G\left(i\Delta x, n\Delta t\right) \tag{1.20}$$

where, Δt is the timestep and $\Delta x = \frac{L}{N}$ the grid width computed from L the length of the cable and N the number of spatial grid points.

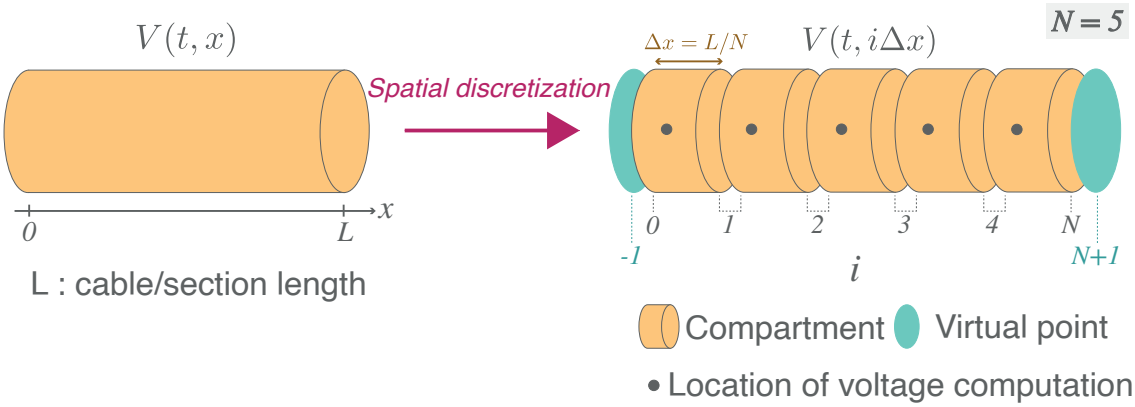


Figure 1.15: Spatial discretization of a section where a cable is approximated as a series of connected cylinders called segments. Virtual points are added at the extremities of the section to verify the no current leak condition.

The membrane potential is then evaluated at the middle of each compartment. The boundary condition that states that no axial current flows at the ends of the cable is respected by adding virtual points at the extremities of the cable.

While the use of explicit methods is suggested to be applicable for multicompartmental model solving according to [Kobayashi et al., 2021], explicit methods remain limited because of the significant constraint imposed by a very small time step for real-time system. In [Kobayashi et al., 2021], the multicompartmental model is using a complex method of higher accuracy that is Runge-Kutta-Chebyshev method with a very small time step but simpler explicit solvers of lower accuracy as the Forward Euler used for the single compartment modeling is known unstable for multicompartmental model [Carnevale and Hines, 2006].

A numerically stable solver appropriated for stiff systems and widely used that will be detailed in this manuscript is the Crank-Nicholson method presented in previous subsection. It relies on an evaluation at half a time step using Backward Euler advanced over the full interval with Forward Euler and is known stable and accurate [Carnevale and Hines, 2006, Hines, 1984]. The equation applied to the membrane potential is equated in Equation 1.21.

$$V_i^{n+1} = 2V_i^{n+\frac{1}{2}} - V_i^n \tag{1.21}$$

The second order correct and numerically stable solution of the finite difference form of Equation 1.21 is expressed in 1.22 as a tridiagonal linear system evaluated at half a time step.

$$L_i V_{i-1}^{n+\frac{1}{2}} + D_i V_i^{n+\frac{1}{2}} + U_i V_{i+1}^{n+\frac{1}{2}} = B_i \tag{1.22}$$

where, L is the lower diagonal, D is the main diagonal, U is the upper diagonal and B the right-hand side of the system defined in Equation 1.23.

$$
\begin{aligned}
L_i &= \frac{1}{2\pi a_i \Delta x} \frac{\pi a_{i-1}^2}{R_a \Delta x} \\
U_i &= \frac{1}{2\pi a_i \Delta x} \frac{\pi a_{i+1}^2}{R_a \Delta x} \\
D_i &= -\left(L_i + U_i + \frac{2C_m}{\Delta t} + g_{tot}^{n+\frac{1}{2}} \right) \\
B_i &= \frac{2C_m}{\Delta t} + g_{Na}^{n+\frac{1}{2}} E_{Na} + g_K^{n+\frac{1}{2}} E_K + g_L^{n+\frac{1}{2}} E_L + \delta_{i0} \frac{I_{inj}(t)}{2\pi a \Delta x}
\end{aligned}
\tag{1.23}
$$

with I_{inj} the current injected and δ_{i0} the Kronecker delta.

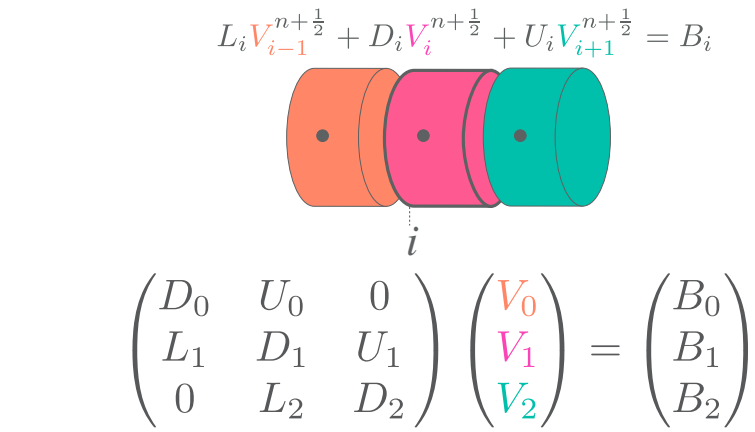


Figure 1.16: Illustration of the tridiagonal systems of equations corresponding to the computation of the membrane potential in a section of a multicompartmental neuron.

The complete structure of the neuron corresponds to a tree of unbranched cables (sections) divided in N compartments, thus adding off-diagonal coefficients to the tridiagonal linear system [Hines, 1984]. Through wise numbering of the nodes in the tree, the tridiagonal matrix resulting is solvable thanks to Hines matrix solver.

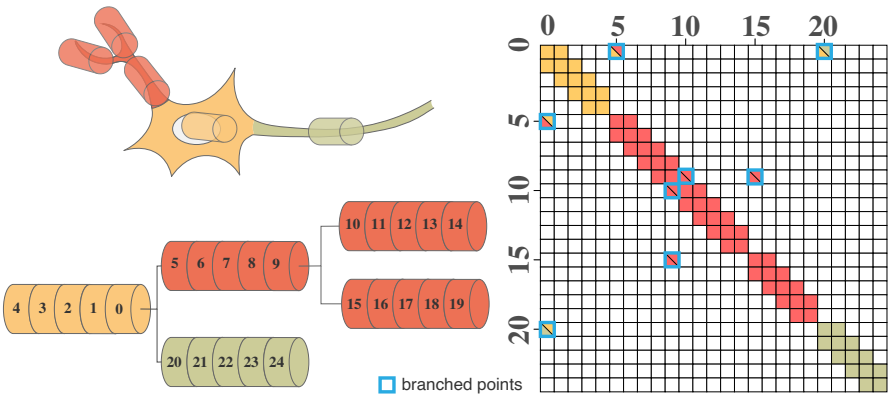


Figure 1.17: Illustration of mainly tridiagonal matrix with sparse coefficients (Hines matrix) at branches points generated by the multicompartmental neuron structure.

The introduction of the spatial dimension in the multicompartmental modeling is translated by a higher complexity requiring complex solver with higher implementation cost, making the implementation of this model more challenging on a digital platform.

1.7 Summary

This chapter introduced the basic notions of biology required to the understanding of the nervous system morphology and its main components. It granted the essential knowledge of how neurons process and send the information through the difference parts like axons and dendrites through synapses. Nonetheless, this complex system may dysfunction because of neurodegenerative diseases already affecting a significant part of the population and only expected to grow. Hence, the treatment of these diseases represents a great challenge requiring joined efforts of various fields to find efficient alternative treatments. This chapter also covers the role and methods of the use of models to study human body and its mechanism, presenting biological models and artificial models both serving the purpose of investigate and understand the complex processes of human body. Particular attention was paid to the artificial model of neurons and their formation in network in view of developing biomimetic system working toward the discovery of alternative treatments such as electroceuticals [Reardon, 2014, Famm et al., 2013]. To conclude, this chapter highlighted biomimetic artificial neuron models and the importance of biological meaningfulness to study neurological disorders through biophysically detailed single compartment or multicompartmental modeling.

2 Introducing the AMD Xilinx SOM K26

2.1 Introduction

As most of integrated circuits, FPGA have considerably evolved and improved by taking advantage of technological improvements in transistor scaling reduction [Boutros and Betz, 2021]. One of the main benefits of FPGA architecture lie in its flexible conception of logical circuits and its highly parallel architecture allowing design of complex real-time applications.

However, the main drawback of the FPGA lied in the communication that often remained limited or complex because of its low-level of description not well suited to the implementation of communication protocols that usually include several complex or abstract layers. Along with the computer power growth [Nordhaus, 2007], other architectures like GPU also considerably improved and widely spreads in various fields [Dally et al., 2021, Nickolls and Dally, 2010].

The co-design approach that existed for example between CPU and GPU also considerably strengthened for general purpose computation [Arora, 2012] and also explored various architecture with SoCs. A great example is the ARM architecture [Seal, 2000] that emerged from the transistor scaling reduction among many other architectures of CPU [Patterson and Hennessy, 2013]. The small size of the ARM cores made it a crucial element of the design of embedded system such as smartphones [PratapSingh and Kumar, 2014].

The integration of processors together with a FPGA on a SoC was widely spread thanks to the ARM architecture has sparked the creation of SoC FPGA such as the Zynq for the manufacturer AMD Xilinx [Crockett et al., 2014]. It allowed compensating for the lack of versatile interfacing of FPGA and allowed the creation of completely embedded and flexible designs thanks to the introduction processors.

This chapter explores the choice of the digital platform to develop a real-time biomimetic SNN on SoC FPGA that would provide flexibility and versatility to adapt and interface various applications. It will also introduce the essential communication protocols and interface that are used in our system. First, the technological context that led to the choice of the selected SoC FPGA will be discussed. Then, the architecture and functionalities of an AMD Xilinx SoC FPGA that is the SOM K26 as well as its characteristics will be explained. Finally, the communication protocols used in the system will be detailed.

2.2 Technological context

To enable bidirectional biohybrid experiments and develop bioelectrical therapeutic solutions for health care like electroceutics [Famm et al., 2013, Reardon, 2014, Di Florio et al., 2023], embedded real-time biophysical interfaces and SNN processing are mandatory to ensure interaction at biological timescale [Sharifshazileh et al., 2021, Corradi and Indiveri, 2015, Mosbacher et al., 2020, George et al., 2020].

A real-time system takes into account the temporal constraints of the system being studied, measured or simulated. Applied to neural networks, real-time behavior reproduces nerve impulses with respect to the biological time.

While the most used solutions for neuron emulation are software based did not show convincing results for real-time computing, other available systems include analog chips that propose optimal design for a very low power consumption. However, analog chips often suffer from a lack of flexibility because of the fixed hardware. The digital platforms then remain a more appropriate platform for our applications making of GPU and FPGA promising choices. Especially, SoC FPGA combines the flexibility, speed and low cost of the FPGA with the versatility and compatibility of the CPU.

This section will present the technological context of the main neural modeling implementation and focusing on FPGA implementation to finally present the selected target for our system.

2.2.1 State of the art of biomimetic models

Most current solutions for biomimetic SNN simulations are software-based running on CPU such as NEURON [Hines and Carnevale, 2001], NEST [Gewaltig and Diesmann, 2007] or Brian2 [Stimberg et al., 2019] tools that show significantly high computation time, especially for complex neuron model with synaptic plasticity.

The Brian software can run on GPU using CUDA, thus greatly improving the computation speed [Stimberg et al., 2013]. Also running on GPU, [Kobayashi et al., 2021] presents the emulation of a HH-based multicompartmental model with 3074 neurons for an average of 674 compartments per neuron forming 780,404 synapses computed in 2.5h for 1 second.

As for Supercomputers as in [Jordan et al., 2018], 18,000 neurons with 1,250 synapses each simulated over 1 second of biological time took 5 minutes of real time. The power consumption was estimated to about 60 to 70 kW per rack for a total of 28 racks. The Japanese supercomputer K was able to simulate 1.74 billion nerve cells and 10.4 trillion synapses in one second of biological time in 40 minutes. The power consumption of these machines remains high and their accessibility low.

In contrast, hardware implementations can perform real-time simulations with low power consumption. Moreover, software-based solutions are not suited for real-time emulation at millisecond time step [Van Albada et al., 2018] contrary to hardware-based SNNs. In the context of biohybrid experiments requiring real-time embedded system hardware implementations appear more relevant.

Hardware implementations of neural networks can be divided into two categories. On the one hand, mixed implementations based on integrated circuit designs; on the other, digital implementations based on FPGAs, microprocessors, microcontrollers or neurochips.

Analogic and mixed implementations. Most of these systems consist of an analog core that simulates the neuron and generally digital circuits for the synapses and plasticity. [Hasler et al., 2007] and [George et al., 2013] implement a multicompartmental model including reconfigurable dendrites. [Sorensen et al., 2004], [Binczak et al., 2006], [Renaud et al., 2007], [Levi et al., 2018] and [Natarajan and Hasler, 2018] integrate conductance-based models. [Liu and Douglas, 2004], [Vogelstein et al., 2004], [Indiveri and Fusi, 2007], [Schemmel et al., 2007], [Qiao et al., 2015], [Kohno et al., 2016] and [Valentian et al., 2019] present threshold-based models.

Digital implementations. The majority of the digital implementations the majority are bioinspired models for computational purposes running on FPGA or neuromorphic chips. [Nazari et al., 2015] shows an application of [Cassidy et al., 2011] using one million of threshold-based neurons. [Wang et al., 2013] implements 4,000 neurons connecting 1.15 million synapses. The prominent SNNs hardware platforms and major projects are Merolla [Merolla et al., 2014], BrainScaleS-2 [Pehle et al., 2022], SpiNNaker [Painkras et al., 2013], SpiNNaker 2 [Mayr et al., 2019] and Loihi [Davies et al., 2018].

SpiNNaker was designed as a part of the neuromorphic computing platform for the Human Brain Project [Amunts et al., 2016]. Its complete design includes ARM cores organised in 10 racks of 100,000 cores each with each core emulating 1,000 neurons, thus aiming to emulate a billion neurons in real time using LIF model [Van Albada et al., 2018].

BrainScaleS-2, that is also part of the Human Brain Project, implements multiples cores of 512 neurons to reach about 4 millions neurons for 880 millions synapses.

The Loihi chip is a neuromorphic chip implementing 130,000 spiking neurons and 130 million synapses.

While some of these systems present mobile versions like [Stradmann et al., 2022] for BrainScaleS-2, they often are not suited for embedded applications.

The state-of-the-art shows a large amount of digital implementations capable of running large number of neurons. However, only few implementations present HH model and fewer multicompartmental model of HH model. This justifies the importance of this work to propose an accessible and flexible HH model implementation.

2.2.2 Follow-up on HH FPGA implementations

As introduced previously, the complexity of analog circuits and the power consumption of supercomputers pushed the use of digital circuits for neural network implementation. Nonetheless, because of its high complexity and implementation cost, the HH model shows few digital implementations and even fewer implementations on FPGA. It is mostly explained by the mathematical operations required by the model such as exponential and divisions that are not suited for FPGA architecture and resource-intensive. The accuracy required also significantly constrains the data coding as FPGA architecture shows resource-intensive floating-point operations. Also, the FPGA implementation of the multicompartmental HH model is rare because of the complexity of the solver required. Hence, the implementation of the HH model requires simplifications.

Single compartment implementations. In the single compartment implementations, [Osorio, 2016] presents a time- and resource-intensive pipeline architecture using floating-point computations and complex methods explicit solving methods such as Runge-Kutta and Goldsmith algorithms paired with Taylor series expansion for complex operators.

[Yaghini Bonabi et al., 2014] shows good computational accuracies using fast and resource-efficient algorithms such as the COordinate Rotation DIgital Computer (CORDIC) and explicit Forward Euler method enabling simulation of 120 neurons connected. [Akbarzadeh-Sherbaf et al., 2018] presented another way of calculating neurons increasing to 5,120 neurons computed in real-time.

While demonstrating numerical implementation techniques to improve the performances, these implementations presented only FS type neurons. Additionally, none of them presents a fast and efficient way to make the system flexible, i.e. to modify biophysical parameters in real time.

A work proposed by our team proposes the implementation of FS, RS, IB and LTS neurons with dynamic parameters tuning and synaptic connection hardware-fixed synapses [Khoyratee et al., 2019] for a total of 500 neurons per calculation core.

Multicompartmental implementations. Multicompartmental HH model implementation on FPGA is almost non-existent, only some experimental results like [Ding et al., 2021] that shows two-compartment neurons and methodology like [Beaubois et al., 2022] exist. Thus making of the implementation presented in this manuscript the first FPGA implementation of multicompartmental HH model on FPGA.

2.2.3 Selected Targets Overview

The main target selected is the Kria K26 SOM from AMD Xilinx embedded on the development platforms Kria KV260 Vision AI Starter Kit and Kria KR260 Robotics Starter Kit advertised as cost-optimized targets. This SoC FPGA is based on the Zynq™ UltraScale™ MPSoC architecture thus featuring processors and FPGA on the same chip. This choice of this target is justified by its capacity of running an operating system and offers good FPGA and processor performances,

versatility and flexibility thanks to its architecture and various interfaces all for an optimized cost. The compact size of the carrier boards also constitutes a considerable benefit to include in a biohybrid experimental setup.

In total the chip incorporates 6 cores with Quad-core Arm® Cortex®-A53 MPCore™ up to 1.5GHz and Dual-core Arm Cortex-R5F MPCore up to 533MHz. It also includes a graphics chipset with the Mali™-400 MP2 up to 667MHz and on-SOM memory with 4 GB of DDR4 memory 16 GB of flash. The FPGA contains 256,000 system logic cells, 26.6 Mb of on-chip SRAM and 1,248 DSP slices. The development boards present various communication interfaces such as USB3.0, SATA 3.1, DisplayPort, Gigabit Ethernet as shown in Figure 2.1.

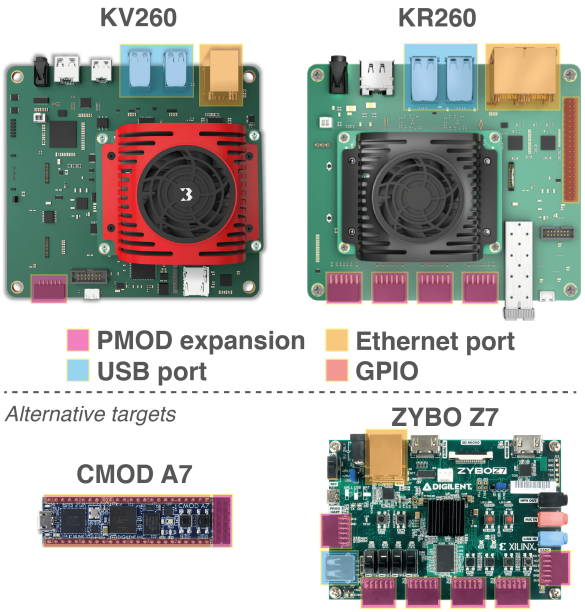


Figure 2.1: Selected development platforms and their main interfaces as well as alternative targets for lower consumption applications.

In order to explore energy-efficient and compact solutions, two others targets were used to implement intermediate or alternative reduced versions of the system: Digilent Zybo Z7-20 and Digilent CMOD A7 presented in Figure 2.1. The Digilent Zybo Z7-20 is also based on a smaller Zynq architecture that features Dual-core Arm® Cortex®-A9 MPCore™ up to 866 MHz with no graphic chipset. The Digilent CMOD A7 is a small target embedding only a FPGA. The Figure 2.1 summarizes the targets selected and the Table 2.1.

Development board	**KR260/KV260**	**ZyboZ7-20**	**CMOD A7**
Target	XCK26	XC7Z020	XC7A35T
Processor	4x ARM Cortex A53 2x ARM Cortex R5F	2x ARM Cortex A9	-
System logic cell	256,000	85,000	33,280
DSP slice	1,248	220	90
On-chip memory (Mb)	26.6	4.9	1.8
Fmax BRAM (MHz)	585	388.2	200
Fmax URAM (MHz)	500	-	-
Fmax DSP (MHz)	644	464.25	464.25

Table 2.1: Comparison of the main characteristics of the selected targets. The maximum frequencies correspond to the maximum frequencies of components in the best case scenario from the datasheet. BRAM corresponds to the on-chip memory blocks.

2.3 Communication Protocols and Interfaces

Communication protocols are rules and standards that define how systems communicate with one another to transmit information. On the other hand, interfaces refer to physical or logical connections between two devices or systems that allows them to communicate with each other.

Communication protocols are key elements in most systems as they guarantee coherency and integrity of data with respect to the latency and throughput imposed by the application.

Most protocols comply with standards covering criteria such as service quality, security or data integrity. Standard also are applied to interface ruling the physical constraints of the connector or their logical connections.

There are numerous communication protocols, often designed to meet a specific need or optimized for certain applications. Hence, choosing a communication protocol requires an analysis of the needs and constraints of the system while also considering the target architecture.

This section aims to introduce a basic knowledge of the essential communication protocols and interfaces involved in the system developed.

2.3.1 AXI Protocol: Efficient SoC Interconnect Communication

The Advanced Microcontroller Bus Architecture (AMBA) is an open-standard, on-chip interconnect specification for the connection and management of functional blocks in SoCs designs. Essentially, AMBA protocols define how functional blocks communicate with each other that simplifies the development of designs with multiple processors and large numbers of controllers and peripherals. It provides several benefits including flexibility to work with a wide range of SoCs, compatibility through standard interface specifications, bandwidth that corresponds to the product of the clock speed and the width of the data bus and relatively low latency in burst-based sytem.

The Advanced eXtensible Interface (AXI) protocol comes with the third generation of AMBA interface defined in the AMBA 3 specification (see Figure 2.2). It is targeted at high performance, high clock frequency system designs and includes features that make it suitable for high-speed submicrometer interconnect.

The AXI4 protocol that comes with the AMBA 4 specifications in 2010 introduced the AMBA 4 AXI4 along with the subsets AXI4-Stream and AXI4-Lite protocols (see Figure 2.2), all used in the system mainly for the interactions between the PS and PL.

The AXI4-Stream protocol is designed for unidirectional data transfers from with reduced signal routing, which is well suited for implementation in FPGAs.

The AXI4-Lite protocol is intended for communication with simpler, smaller control register-style interfaces in components.

The documentation of the manufacturer [Xilinx, 2017] describes the three types of AXI4 interfaces as:

- **AXI4**: for high-performance memory-mapped requirements
- **AXI4-Lite**: for simple, low-throughput memory-mapped communication (for example, to and from control and status registers).
- **AXI4-Stream**: for high-speed streaming data

Both AXI4 and AXI4-Lite interfaces consist of five different channels: read and write address channels, read and write data channel as well as write response channel. Data can be transferred in both directions between the master and slave simultaneously with varying data

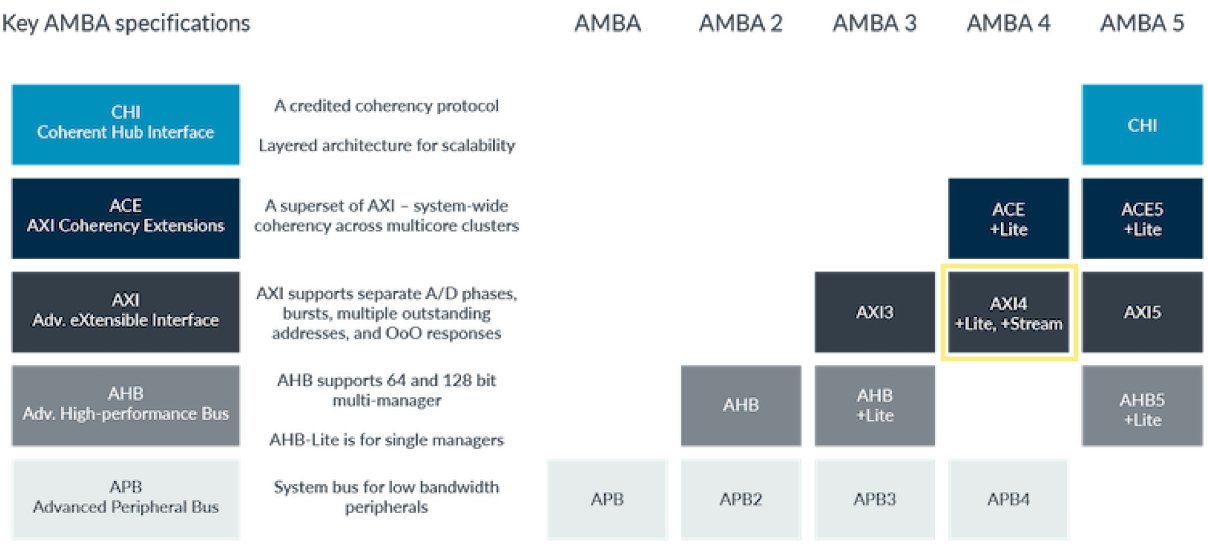


Figure 2.2: Evolution of Advanced Microcontroller Bus Architecture (AMBA) specifications to meet the demands of processors and new technologies. The yellow rectangle shows the AXI4 standard used in the system.

transfer sizes. The AXI4 is limited to burst transactions of up to 256 data transfers where the AXI4-Lite does not allow burst and only allows one data transfer per transaction. At a hardware level, AXI4 allows systems to be built with a different clock for each AXI master-slave pair, a convenient feature for FPGA implementation. The Figure 2.3 shows write and read transactions using the AXI protocol that provides separate data and address connections for reads and writes, which allows simultaneous and bidirectional data transfer.

Read transaction. The master initiates a read through the read address channel by specifying the size of the transfer and the address. The slave transfers the data using the read data channel.

Write transaction. The master initiates a write through the write address channel by specifying the size of the transfer and the address, then transfers data through the write data channel. The slave indicates the status of the write transaction on the write response channel.

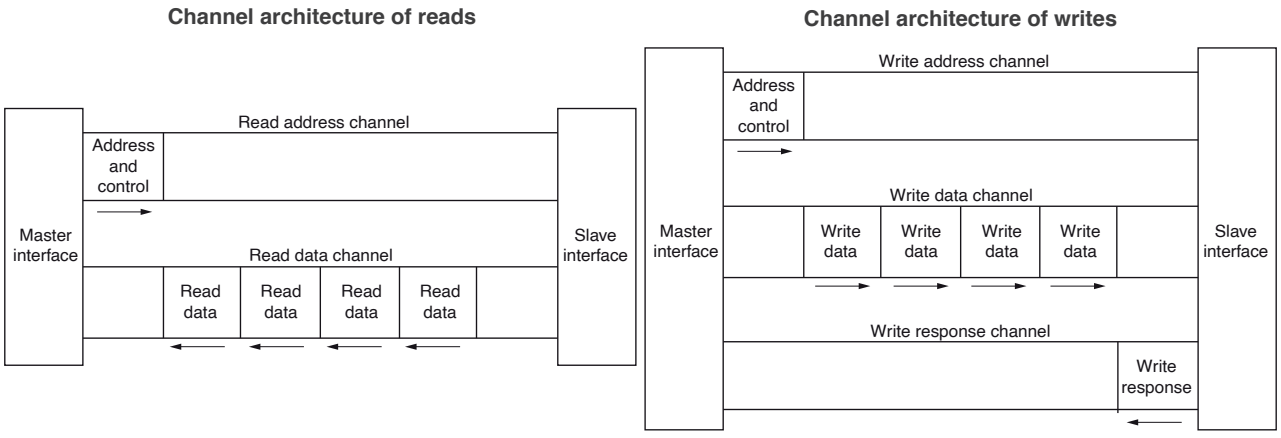


Figure 2.3: Architecture of read and write channels in AXI protocol that shows AXI4 read and write transactions.

2.3.2 USB: The Universal Standard for Device Integration

The Universal Serial Bus (USB) protocol is a standard communication protocol enabling devices to connect and communicate with each other. USB is widely used for connecting peripherals such as mouses, keyboards, printers, etc. It is a standard yet flexible protocol allowing communication with a wide range of devices through the use classes adapting to the size and functionalities of the device as shown in Figure 2.4. In addition, it shows high transmission speed up to 10 Gb/s in its latest standards while also providing other features such as power supply or error detection. The USB interface have many connector types (type A, type B, micro, mini, ...) as well as cables. USB 2.0 and USB 3.0 are two different generations of USB specification offering different capabilities and improvements on performances such as throughput.

USB is based on two primary roles that devices can play when communicating over a USB connection. The host initiates and controls the communication on the USB bus. It is typically responsible for tasks such as managing the USB topology and addressing of devices, initiating communication sessions or sending and receiving data to and from devices. The device is a peripheral or a gadget that communicates with the host based on its specific class that defines its functionality and interactions. Typically, the device respond to commands and transfer data to and from the host as requested.

As the USB is a complex protocol, its implementation on the FPGA can be challenging and relies on the support available in terms of IP cores. USB has multiple layers and intricacies that involves low-level, thus implementing the entire USB protocol can be demanding and time-consuming forcing the use of dedicated IP cores.

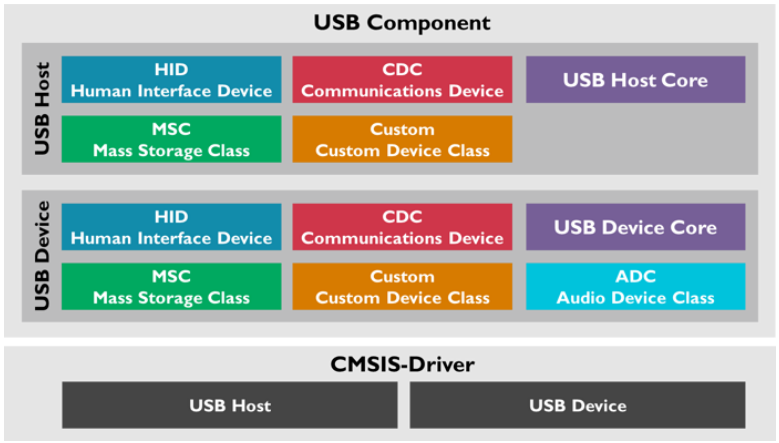


Figure 2.4: Structure of the Universal Serial Bus (USB) that shows the device classes as well as the host and device roles.

The USB protocols is used including various classes in the system as it is handled by a generic operating system. Nonetheless, an intermediate version of the system explored communication through USB2.0 using the Communication Class Device (CDC) as the main data communication protocol.

2.3.3 Ethernet: Connecting Devices in Networks

Ethernet protocol dictates the way devices communicate within local and wide area networks. By utilizing a wired physical medium, typically Ethernet cables, it enables data exchange between computers, servers, and networking equipment. It is ubiquitously used in homes, businesses, data centers and the broader internet infrastructure.

Ethernet operates on the principle of packets, in other words, the data is broke down into packets for efficient transmission and routing. The protocol offers various speeds, ranging from 10 Mb/s Ethernet to modern variants like 1/10/100 Gigabit Ethernet (1 Gb/s, 10 Gb/s or 100 Gb/s). Ethernet has a deterministic nature that allow to predict or determine its behavior with certainty as well as low latency, thus making of Ethernet a solution suited for real-time applications such as streaming and online gaming. Ethernet is a complex protocol and interface that features several layers illustrated in Figure 2.5.

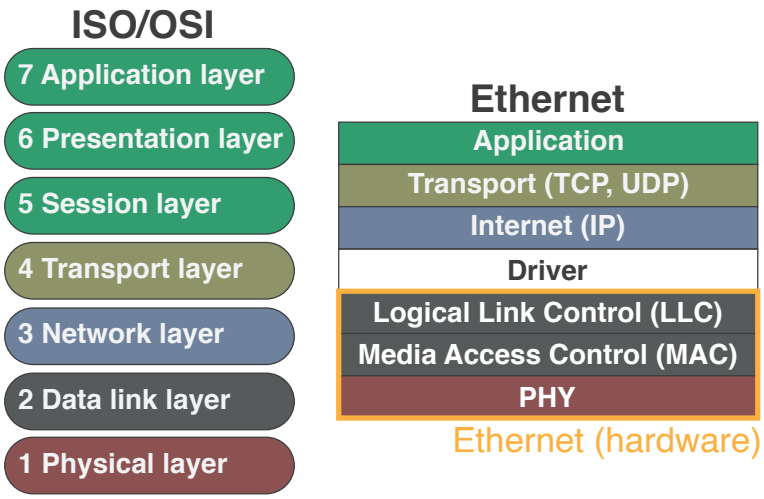


Figure 2.5: Open Systems Interconnection model (OSI model) is a conceptual model from the International Organization for Standardization (ISO).

Implementing Ethernet on FPGA involves challenges due to its complexity and multiple layers that are difficult to translate at the low-level of description proposed by FPGA, forcing the use of dedicated IP cores. FPGA implementations of the Ethernet protocol further enhance the development of real-time applications with high rates and low latency thanks to the FPGA architecture that does not include software that induces fluctuating latency.

Nonetheless, its development is rather difficult and time-consuming without the use of dedicated IP cores. The use of the Ethernet protocol on a processor-based architecture also shows great performances thanks to the use of components that optimize data transfer like DMA that will be detailed in the next section.

The Ethernet communication is used in the system as the main data communication protocol thanks to its high throughput and low latency. It is handled on a software side by a generic operating system, thus greatly simplifying the implementation.

2.3.4 UART: Basic Device Communication

The Universal Asynchronous Receiver/Transmitter (UART) protocol is a serial communication protocol that transmits data between peripherals such as computers and microcontrollers. It is widely used in numerous applications amongst communication with microcontrollers or data communication between peripherals in RFID based applications or GPS modules.

The UART protocol uses asynchronous serial communication with configurable speed. An asynchronous communication implies that no clock signal synchronize the output bits from the transmitting device going to the receiving end. The UART contains two signal: transmitter (TX) and receiver (RX). The data is transferred bit by bit as shown in Figure 2.6.

UART is a simple protocol that can easily implemented on FPGA with a low implementation cost, but that features low throughput and limited reliability in cables.

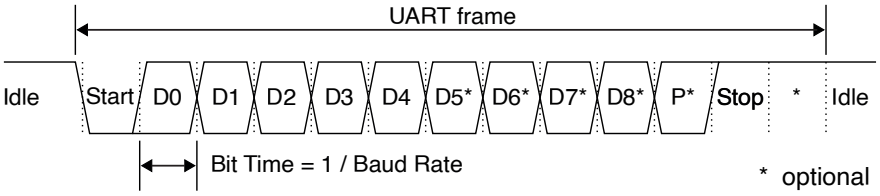


Figure 2.6: Elements of a UART frame showing start, stop, parity (P) and data bits (D).

The UART is used in almost all the versions of system in both hardware and software as a communication protocol forwarding debug data.

2.3.5 SPI: Simplified Device Connection

The Serial Peripheral Interface (SPI) protocol is a serial communication protocol usually used to connect microcontrollers and peripherals. Just like the UART protocol, SPI is widely employed in a diverse range of applications including interfacing with microcontrollers.

Unlike UART, the SPI protocol is a synchronous serial communication that shares the clock signal between devices for synchronization. The SPI interface uses multiple lines for communication: Master Out Slave In (MOSI), Master In Slave Out (MISO), Serial Clock (SCK) and Chip Select (CS). The data is transferred bit by bit when the chip select signal (CS) is set low as depicted in Figure 2.7.

SPI allows transfers at higher speed than UART and simultaneous communication with multiple devices. While it requires more wiring compared to UART, SPI is a protocol easily and efficiently implementable on FPGA that comes at a low implementation cost.

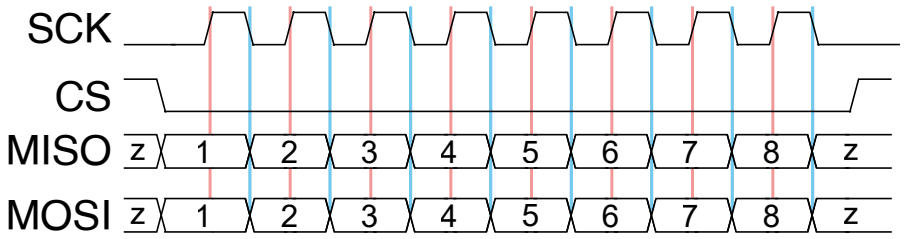


Figure 2.7: Elements of a SPI frame showing the different signals involved.

The SPI protocol is widely used in the system for the interactions between the hardware and other external components connecting with the system.

2.3.6 Wi-Fi: Wireless Device Connectivity

The Wi-Fi (a brand name standing for Wireless Fidelity) protocol allows devices to connect and communicate wirelessly, eliminating the need for physical cables. Wi-Fi is widely used for creating wireless local area networks (LANs) that provide internet access to devices like smartphones, laptops and IoT gadgets.

Wi-Fi operates by transmitting data using radio waves in specific frequency bands, typically 2.4 GHz or 5 GHz. Devices equipped with Wi-Fi capability can establish connections to access points (routers) and communicate with each other. On the technical side, the IEEE 802.11 standard defines the protocols that enable communications with current Wi-Fi-enabled wireless devices. The standards operate on varying frequencies, deliver different bandwidth and support different numbers of channels. A Wi-Fi frame has a rather complex structure and multiple layers including various related to network with a maximum size of 2346 bytes. Typically, the latency

observed with Wi-Fi is about 5 to 10 ms for a throughput that can reach a theoretical 10 Gb/s in its latest theory.

As Wi-Fi is a complex protocol composed of multiple layers, its implementation remains highly challenging on FPGA. Even with dedicated controller handling the physical layer (PHY), the entire protocol stack includes higher-level protocols that are not well-suited for FPGA that operates close to the hardware with a low level of description.

The Wi-Fi communication is used in the main system as well as in an alternative version as an optional communication protocol for the main data, thus providing an interesting solution for embedded applications thanks to its wireless nature.

2.3.7 PMOD: Standard Device Interfacing

The Peripheral MODule interface (PMOD) standard defined by Diligent is a versatile and widely adopted interface for seamlessly connecting peripheral modules to microcontroller and FPGA-based systems. PMOD offers a standardized approach to extending system functionality, allowing a diverse range of peripherals like sensors, actuators and communication modules to be easily integrated.

The PMOD connects peripheral modules to FPGA and microcontroller development boards using either 6 pins (power, ground and 4 signals) or 12 pins (2 powers, 2 grounds and 8 signals) as illustrated in Figure 2.8. PMOD may also refer to modules compatible with the PMOD interface.

Thanks to its reconfigurable nature, FPGA are very flexible system that can adapt to a wide range of hardware and allow the design of custom interfaces. Using a standard interfacing through PMOD allows facilitated communication between PMOD modules integrating peripherals and FPGA. Thanks to their plug-and-play nature, PMODs allow rapid prototyping and embedded systems development, thus enabling adaptable expansion of FPGA capabilities across a wide range of applications. The PMODs largely used by the team include the PMOD DA4 that integrates a 12-bit digital to analog converter to display analog waves and the PMOD ESP32 that integrates an ESP32 microcontroller featuring Wi-Fi and Bluetooth.

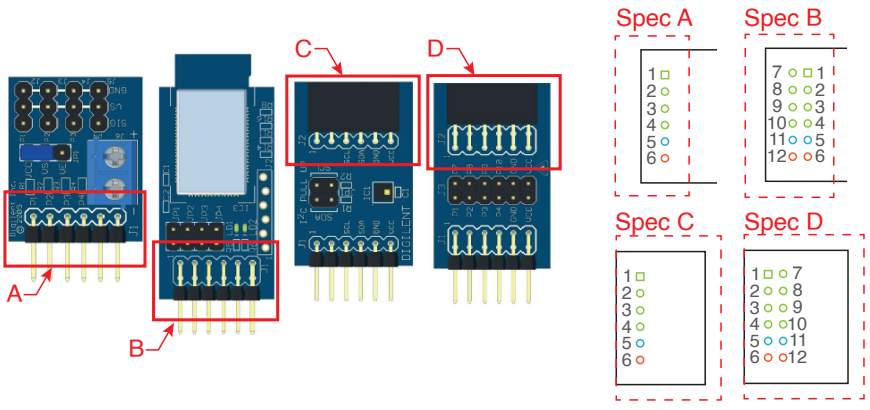


Figure 2.8: Peripheral MODule interface (PMOD) specifications.

The PMOD are largely used in the system through the interfaces available on all the targets to provide a generic utilization of various components such as DAC and Wi-Fi communications for the monitoring of the system.

2.4 Architecture and components of the SOM K26

The SOM K26 is a custom-built Zynq™ UltraScale™ MPSoC that is a family of integrated circuits developed by AMD Xilinx which combines the processing capabilities of processors with programmable logic on a single chip. It is designed to provide a high level of processing performance, flexibility, and integration for a wide range of applications, particularly in embedded systems and high-performance computing. The Figure 2.9 recapitulates the different blocks and components of the Zynq™ UltraScale™ MPSoC architecture.

The chip is then divided in two parts: Programmable Logic (PL) and Programmable Logic (PL). The PS part that is the ARM-based processing system consists of multiple cores, including Cortex-A53 application processors for general-purpose computing tasks and Cortex-R5 real-time processors for time-critical tasks. These cores can run different operating systems and handle various applications and operating modes. The PL is the programmable logic portion that consists of reconfigurable logic cells and resources, i.e. the FPGA. In the standard operation, it is designed to be used as hardware accelerator thanks to various components of the FPGA like Digital Signal Processor (DSP) slices.

The on-chip interconnect allows efficient communication between the PS and PL to allow data exchange the software and hardware components through various methods.

It features various memory components including DDR4 memory controllers and on-chip memory as well as various cache levels. It also features various interfaces such as PCIe, USB, Ethernet, and various other standard I/O interfaces to connect with external devices and networks. This section aims to provide basic understanding of the different parts and components.

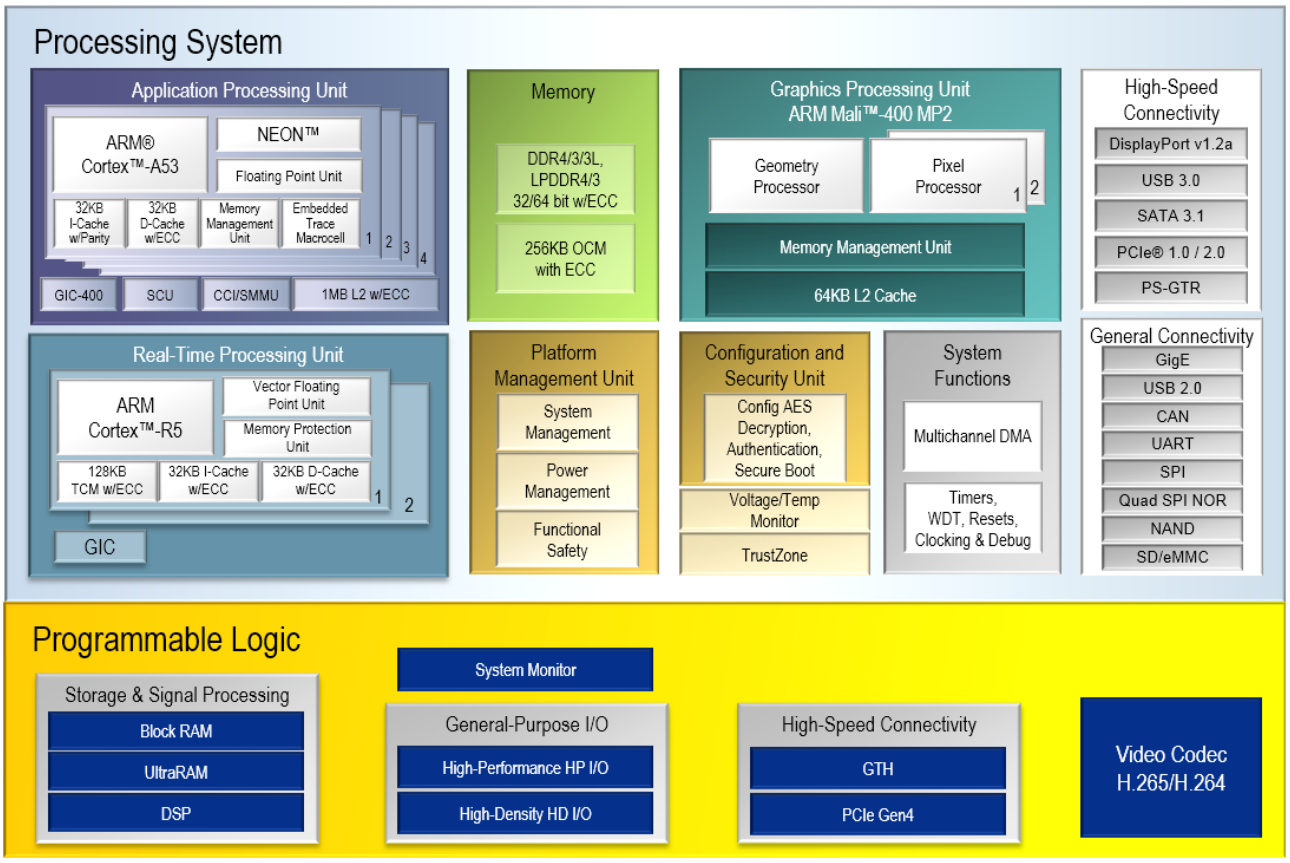


Figure 2.9: Architecture block diagram of the Zynq™ UltraScale™ MPSoC devices of EV variant adapted to create the SOM K26 that features quad application processor and GPU.

2.4.1 Application Processing Unit (APU)

The Application Processing Unit (APU) consists of four Cortex-A53 MPCore processors, L2 cache and related functionality designed for system control and compute-intensive applications that do not need real-time performance (see Figure 2.9). The Cortex-A53 MPCore processor is the one of the most power-efficient Arm v8 processor capable of seamless support for 32-bit and 64-bit code. It makes use of a highly efficient pipeline with advanced fetch and data access techniques for performance. It fits in a power and area footprint suitable for entry-level devices and is at the same time capable of delivering high performance in scalable enterprise systems by integrating several core thanks to a high core density.

The core includes advanced Single-Instruction Multiple-Data (SIMD) and floating-point extension tailored for media and signal processing applications thanks to instructions targeting tasks such as audio, video, 3D graphics, image and speech processing. It also implements the Arm generic timer architecture, debug architecture as well as an external generic interrupt controller.

To put it briefly, the APU integrates powerful cores capable of performing efficiently most calculations. While providing essential features to get closer to a real-time behavior like timers and interrupts, APU is not designed for real-time application and are oriented toward the execution of an operating system.

2.4.2 Real-Time Processing Unit (RPU)

The Real-Time Processing Unit (RPU) includes a pair of Cortex®-R5F processors for real-time processing that implements the Arm v7-R architecture and includes a floating-point unit (Arm VFPv3 instruction set) as shown in Figure 2.9. In the Cortex-R5F processor, the interrupt latency is kept low, achieved by having a dedicated peripheral port that provides low latency access to the interrupt controller and by low-latency memory (tightly-coupled memories). The Cortex-R5F processor is used for many safety-critical applications where the timing is important.

The Cortex-R5F processor is a mid-range CPU for use in deeply-embedded real-time systems. It includes a technology that optimizes the code density and processing throughput to maximize performances. The processor has tightly-coupled memory (TCM) ports for low-latency and deterministic accesses to local RAM in addition to caches for higher performance to general memory. It also supports floating-point arithmetic as well as error checking and correction to provide improved reliability and safety.

To recapitulate, the RPU integrates a core slightly less powerful than the APU, but that is capable of performing efficient calculations at low-latency to guarantee real-time operation. Hence, the use of the core are tailored for real-time applications where computations are to be performed in a given time.

2.4.3 Graphics Processing Unit (GPU)

The Graphics Processing Unit (GPU) is a 2D and 3D graphics subsystem based on the Arm Mali-400 MP2 hardware accelerator (see Figure 2.9). It contains components such as one geometry processor and two pixel processors that perform tasks such as scaling, rotating, and positioning the geometry of objects in the scene or rendering the pixels to produce an image. The GPU can be configured using the software libraries such as openGLES 2.0 API.

Briefly, the GPU embedded is mostly suited for 2D or 3D graphics rendering more than general purpose computation and more particularly in this case to handle the desktop interface of an operating system and its applications.

2.4.4 Memory

Memory is an essential component that involves the notion of access latency —referring to the time it takes to retrieve or store data, throughput —representing the rate of data transfer as well as size of the memory. Most systems integrate multiple memory types to fulfill the different requirements of the system and serve different purposes by having memories with different size and access latencies.

The SOM K26 integrates various types of memory serving different purposes: the processors caches, the external high-speed dynamic random-access memory (DDR DRAM), internal on-chip memory (OCM), tightly-coupled memory (TCM), eMMC, EEPROM, QSPI flash memory and the PL memory blocks.

Processor cache. Processor cache is a special high-speed memory from which processors access their instructions and data. They are small memories (1 MB for L2 cache memory of the APU) that are present close to each core to allow fast access to memory with low latency.

External DDR DRAM. The external memory corresponds to 4 GB 64-bit DDR4 memory that is high-speed dynamic random-access memory mostly accessed by the cores. The most common use is to act as a temporary memory bank for the operating system operation. It is a memory of a large size and high throughput but showing a high latency and that requires frequent refresh operations.

Internal on-chip memory. The on-chip memory corresponds to 256 KB of RAM divided in four banks of 64 KB that is designed to ensure low memory access latency for the RPU.

TCM. Tightly-coupled memory (TCM) is also a low-latency memory used by the RPU. Each Cortex-R5F processor contains two 64-bit wide 64 KB memory banks of TCM memory for a total of 128 KB of memory.

eMMC. The Embedded MultiMediaCard (eMMC) is a 16 GB non-volatile memory integrated in the SOM that retains stored information even after power is removed. It is mostly designed to store the data necessary for the system to operate.

EEPROM. The electrically erasable programmable read-only memory (EEPROM) is a 64 Kb non-volatile memory pre-programmed during manufacturing and that provides device configuration, identification, and manufacturing data.

QSPI flash memory. The Quad SPI flash memory of 512 Mb (64 MB) is also a non-volatile memory that can be used to program the device on startup. Because of its smaller size, it is mostly used to contain the essential data necessary for the system to start.

PL on-chip memory. The on-chip memory in the PL part corresponds to 26.6 Mb of memory organized in block of RAM (BRAM 36 Kb and URAM 288 Kb). As this memory is accessed in hardware at the lowest level by the FPGA, it shows an extremely low latency of one or two clock cycles.

On the development boards KR260 and KV260, a SD card slot provides another memory through the SD card that provides non-volatile memory ranging in the tens of gigabytes. This memory can be used either to store data in files or to store the essential data required for the system to operate.

2.4.5 Connectivity

The connectivity on the SOM contains various interface controllers that can be configured for the applications. Among them can be found Serial Advanced Technology Attachment (SATA) to connect storage devices, DisplayPort to connect display peripherals, PCIe an expansion bus standard for connecting to one or more peripheral devices like a computer and Ethernet SGMII to connect to a network. It also includes four 10/100/1000 tri-speed GEM peripherals that allow controlling Ethernet connection at different speeds. USB 3.0 and USB 2.0 controllers to connect to a wide range of devices and systems also are available as well as two UART (up to 1 Mb/s).

As for the interfaces available in the development boards selected shown in Figure 2.1, the KR260 is the most complete carrier board that integrates 4 Ethernet ports, 4 USB ports, 4 PMOD connectors, DisplayPort, HDMI and GPIO (organized in Raspberry Pi header). The KV260 features fewer connectors with only one Ethernet port and one PMOD connector but for a lower price. The ZyboZ7-20 features 6 PMOD connectors, one Ethernet port and one USB port as well as two HDMI ports while the CMOD A7 integrates only one PMOD connector and GPIOs.

2.4.6 System functions and management

The system functions and management of the SOM K26 include many components and processes that are not essentially necessary to the understanding of the system so as this section will focus on a crucial component that is the DMA. Other components for example include the system monitor that allows monitoring of the temperature of the cores, fundamentally a crucial element to ensure correct functioning of the system but not essential to the understanding of the system developed.

The Direct Memory Access (DMA) is a component tailored to perform memory to memory and memory to I/O buffer transfers. It is for example used to perform data transfer between RAM and a peripheral device such as Ethernet controller. The benefit of the DMA is to offload the CPU and then allow high throughput data transfers without involving the CPU.

Without the DMA, the CPU would issue commands to read data, wait for the data to be fetched and then transfer the data by its self.

With DMA, the CPU sets up the parameters for the data transfer (source, destination, and size) and then hands control over to the DMA controller. The DMA controller directly accesses data and transfers it without the constant involvement of CPU. Once the transfer is complete, the DMA controller notifies the CPU. Meanwhile, the CPU can continue executing other tasks, thus improving overall system performance.

The Figure 2.10 shows the block diagram of the DMA. The DMA is composed of 3 major blocks: the common buffer, the arbiter (AXI write channel and AXI read channel) and the DMA engine channels.

Common buffer. The common buffer is shared between the DMA channels to hold the AXI read transaction data before it goes out on an AXI write channel and is sized to allow utilization of full AXI bandwidth.

Arbiter. Each DMA channel has two AXI read interfaces: one interface is used for reading data buffers and the other interface is used for reading buffer descriptors that contain information about the buffer. The DMA channels share an AXI write channel. The DMA implements round-robin arbitration so as each channel is given an equal portions and in circular order without priority.

DMA engine channels. The DMA channel is responsible for the bulk of the DMA operation and management that are transfer execution, coordination of transfer parameters, monitoring progress and generating notifications for the CPU.

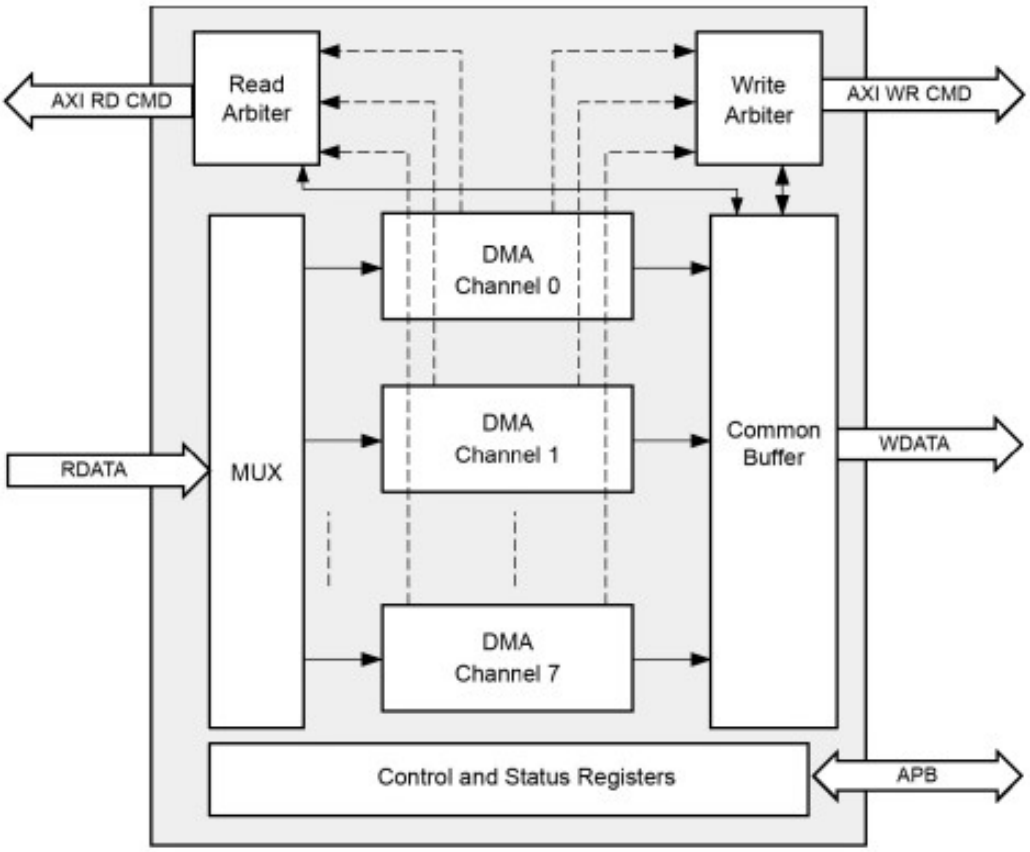


Figure 2.10: Block diagram of the Direct Memory Access (DMA) in Zynq UltraScale+ devices showing the three major blocks: common buffer, arbiter and DMA engine channels.

To put it briefly, the DMA is a crucial component that allows large data transfers between various peripherals of the system with a limited involvement of the CPU, thus significantly improving the performances of the system for data transfers. To illustrate this point with a practical example, the DMA can be used to move data efficiently between the PL and PS part or between the external peripherals like Ethernet to either the PS or PL part.

2.5 Processing System (PS): Embedded Processors within SoC FPGA

The Processing System (PS) contains the Application Processing Unit (APU), Real-Time Processing Unit (RPU), and peripherals. It is often referred to as the software part of the design. The software stacks available for Zynq UltraScale+ MPSoC devices are: bare-metal, Linux and FreeRTOS. Figure 2.11 recapitulates the three software development stacks. This section focuses on the different operating modes as well as the methods used to design each solution.

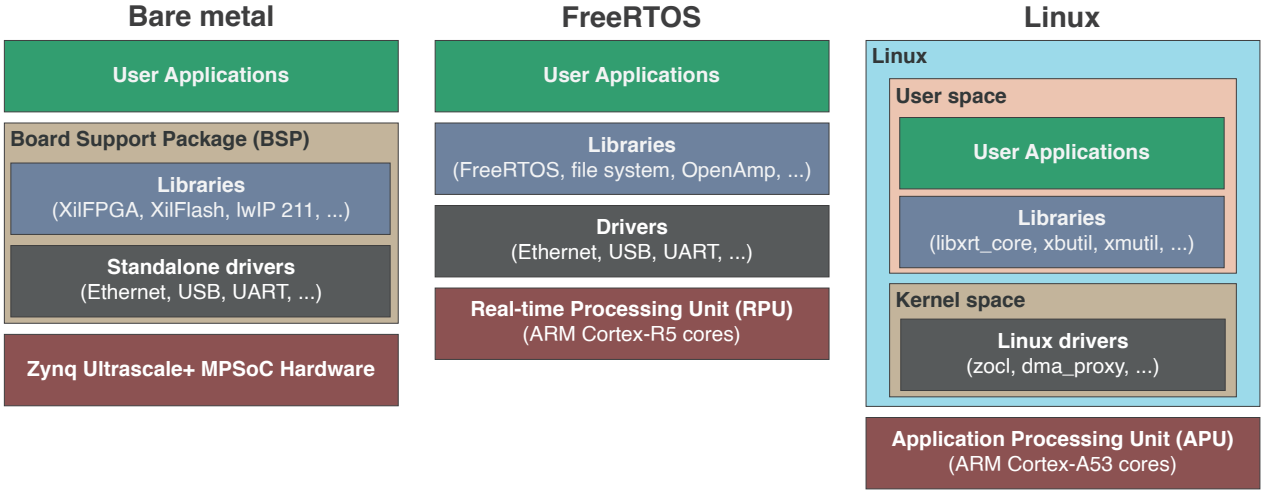


Figure 2.11: Software development stack for bare metal, FreeRTOS and Linux on Zynq MPSoC architecture.

2.5.1 Bare Metal: Low-Level Hardware Programming

Bare metal refers to a software development approach where the code runs directly on the hardware without any underlying operating system (OS), thus directly controlling and utilizing the hardware resources such as the APU or RPU processors, memory and FPGA in the particular case of the Zynq MPSoC architecture. AMD Xilinx provides a bare metal software stack called the standalone board support package (BSP) as part of the Vitis™ software platform (see Figure 2.11). The Standalone BSP gives simple single-threaded environment that provides basic features such as standard input/output and access to processor hardware features.

The bare metal development for Zynq MPSoC consists of code in languages C or C++ compiled through the Vitis tool chain to generate a *.bin* file containing all the components necessary for the system to operate (software executable, fpga configuration, ...). It involves setting up the initialization functions, managing interrupts and interfacing directly with peripherals and memory through low-level functions.

Bare metal programming is suitable for applications that require maximum performance and minimal latency as the software layer is minimal and do not induce fluctuating latencies as with an operating system. It is often used in scenarios such as real-time control, signal processing and custom accelerators using the FPGA fabric. However, this approach requires a solid understanding of the hardware architecture and low-level programming concepts. It is also complex, highly time-consuming and lack versatility because the code is specific to the application developed.

In the process of designing a flexible and user-friendly design, the lack of flexibility and high development time make it less suitable for our application. Nonetheless, the high performances

provided by a bare metal approach were explored in an intermediate version.

2.5.2 FreeRTOS: Real-time operating system

FreeRTOS is a real-time operating system (RTOS) that provides a layer of abstraction that enables efficient management of tasks and scheduling to create a multi-threaded environment. AMD Xilinx includes FreeRTOS support as part of Vitis™ software platform, offering a similar development approach to bare metal but with the functionalities of a real-time operating system like multi-tasks processing. FreeRTOS is aimed to run the RPU of the SOM as shown in Figure 2.11.

FreeRTOS facilitates a multi-threaded environment, enabling to create tasks that run concurrently with specific priorities, manage communication between the tasks and shared access to resources using features like semaphores and mutexes. The layer of abstraction provided by RTOS reducing the complexity of hardware interaction and resource management.
Similarly to bare metal, it involves writing code in C or C++ that uses FreeRTOS libraries for task creation, synchronization, and memory management. The code is also compiled through the Vitis tool chain to generate the board configuration.

FreeRTOS offers a good compromise between performances and versatility as it allows maintaining real-time performances with a certain level of abstraction. However, even though FreeRTOS provides a layer of abstraction, it is still considered as low-level and implies a solid understanding of the hardware architecture and low-level programming concepts.

While FreeRTOS is a solution perfectly suiting the needs of our application, it lacks of flexibility and the low level of abstraction induces high development time. Hence, the sole use of FreeRTOS for the application would not be sufficient. An optimized design, considered but not developed, would lie in both FreeRTOS running on the RPU and a Linux running on the APU.

2.5.3 Linux: Versatile Embedded System Platform

Linux is a well-known and widely used open-source operating system that provides a versatile environment that offers a wide range of features and capabilities for embedded systems with a higher level of abstraction than FreeRTOS. It offers a rich set of drivers, libraries and tools that facilitate efficient interaction with the hardware components including the APU cores and FPGA (see Figure 2.11). The Linux software stack can be leveraged by different tools:

- **PetaLinux Tools**: The PetaLinux tools include tools (Linux source tree, U-Boot and Yocto-based tools) to easily build complete Linux images (kernel, root file system and device tree) and applications for AMD Xilinx devices.
- **Open Source Linux and U-Boot**: The Linux Kernel sources needed for Zynq UltraScale+ MPSoC are provided by AMD Xilinx then allowing the generation of linux images using the open source tools for Linux iamge creation.
- **Commercial Linux Distributions**: Some commercial Linux distributions like Canonical Ubuntu feature support for Xilinx UltraScale+ MPSoC devices including advanced tools for Linux configuration, optimization, and debug.

Linux-based development for the Zynq MPSoC involves configuring and customizing the Linux kernel to match the hardware configuration. This notably includes selecting the necessary

drivers, enabling specific features and tuning the kernel to optimize performance. Additionally, Linux offers a user-space environment that allows to create and manage user applications using familiar programming languages (Python, C, C++, ...) and development tools (SSH, ...).

The benefits of Linux in embedded systems is the vast ecosystem of software packages and libraries, enabling rapid and versatile application development while reducing development time thanks the high level of abstraction. Moreover, Linux allows multitasking and multi-threading capabilities with high level of abstraction making it simple, along with a network stack facilitating connectivity.

However, while Linux offers versatility and convenience, it introduces a layer of complexity that impacts real-time performance. The scheduling and resource management of the operating system introduce fluctuating latencies that makes it less suitable for applications requiring strict real-time behavior. While using a preemptive Linux kernel can mitigate scheduling latencies and enhance real-time capabilities, it remains less suitable than FreeRTOS for strict real-time behavior.

Because Linux brings a powerful and versatile environment and offers features and tools for fast and flexible development, a commercial Linux distribution was used in the system. It allows a flexible and user-friendly monitoring of the system with acceptable latencies, albeit fluctuating.

2.6 Programmable Logic (PL): FPGA Technology within SoC FPGA

The Programmable Logic (PL) contains the Field Programmable Gate Array (FPGA) fabric. It is often referred to as the hardware part of the design. This section aims to introduce the elements and processes that are specific to the FPGA technology to provide the basic knowledge necessary to the understanding of the hardware design of the system.

2.6.1 Data encoding

Data encoding is the process of encoding information or data into a certain format, representation or structure. It is an important element of a system as it directly impact the memory by its size, resource consumption and performances by the complexity of its handling and operations. In artificial neuron modeling, the main data are variables relating to biological property and equation so as it is mainly decimal numbers. Various representation of decimal number exist, but they can be categorized in two categories: fixed-point coding and floating-point. Fixed-point coding and floating-point are both used in the system developed. The Figure 2.12 illustrates the two representations by showing the number of bits used and what they are coding.

Fixed-point coding. Fixed-point coding is an encoding for fractional numbers using a fixed number of bits to code both integer and decimal part of the number additionally with a sign bit, but it basically remains an integer. This data encoding is widely to represent digital data in digital systems such as computers and embedded systems. Fixed-point coding fixes the number of bits used to represent integer and decimal part of the number therefore impacting the accuracy, the more bits used the more accurate the coding is. The main benefit of fixed-point coding lies in low implementation cost and low complexity compared to coding such as floating-point. Therefore, fixed-point coding often suits well embedded systems where resources are limited thanks to its lower memory consumption, simpler and faster computation. Its limitation lies in the operations that imply high magnitude variations like in the case of an addition of very large number with a very small number, so as depending on the encoding size the small number

impact might not be considered.

Floating-point coding. Floating-point coding is an encoding representing fractional numbers using a variable number of bits. Unlike fixed-point coding, floating-point allows a better accuracy by allowing the decimal point to "float". It is done by introducing an exponent part that represents the integer power by which the fraction will be multiplied by (see Figure 2.12). It is mostly used in scientific calculation or high performance applications where high accuracy is required. This representation is more flexible than fixed-point therefore allowing to represent a wider range of numbers with higher accuracy. Indeed, floating-point coding deals very well with high magnitudes variations in operations like multiplications and divisions but a bit less with additions and subtractions. Nevertheless, floating-point coding shows higher complexity and implementation cost on most architectures.

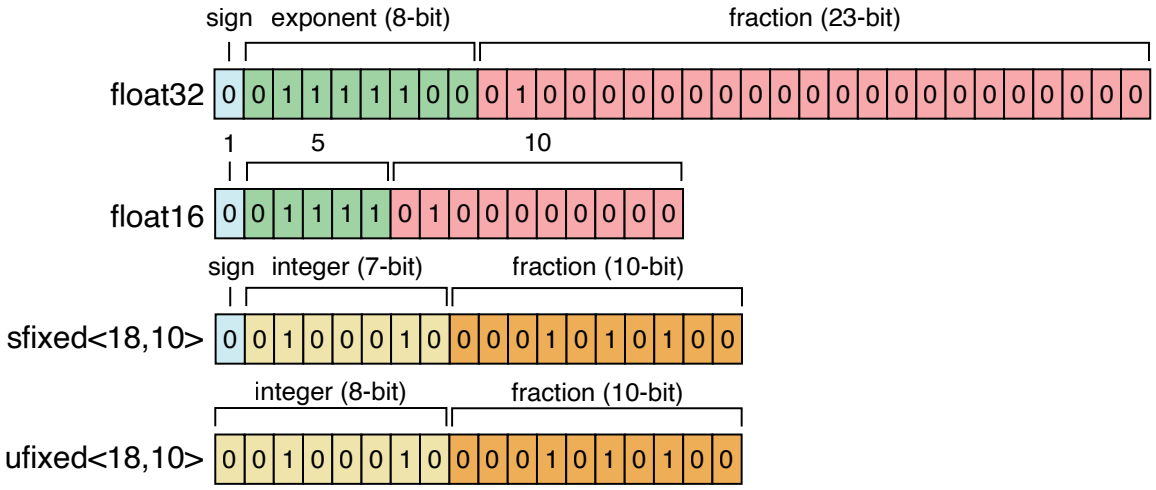


Figure 2.12: Data representation for fixed-point and floating point coding.

2.6.2 High Level Synthesis tools

High Level Synthesis (HLS) is a design process allowing to generate RTL code for FPGA from high-level description such as algorithms or models. As the FPGA development usually involves a low level and time-consuming description language, HLS is an efficient tool to design systems while allowing developers to concentrate more on algorithms and system functionalities rather than hardware related issues. Thus, it can allow a reduction development time and an expansion of the range of developers targeting FPGA architecture.

At the present time, notable supported languages include the widely used C/C++ and MATLAB. While being a recent tool, HLS has proven to be able to generate efficient and reliable implementations thanks to its ability to optimize the resources of the FPGA.

AMD Xilinx proposes the Vitis™ HLS software that is designed to work along with the tool chain of FPGA development provided by AMD Xlinx as illustrated in Figure 2.13. In recent AMD Xilinx targets, HLS is gaining a larger role, especially because of the increasing size of the target that make it more and more difficult for a human to optimize designs. Still in recent versions of AMD Xilinx softwares, a new compiler emerged, v++, that can compile HLS module into PL kernel objects to link with Linux.

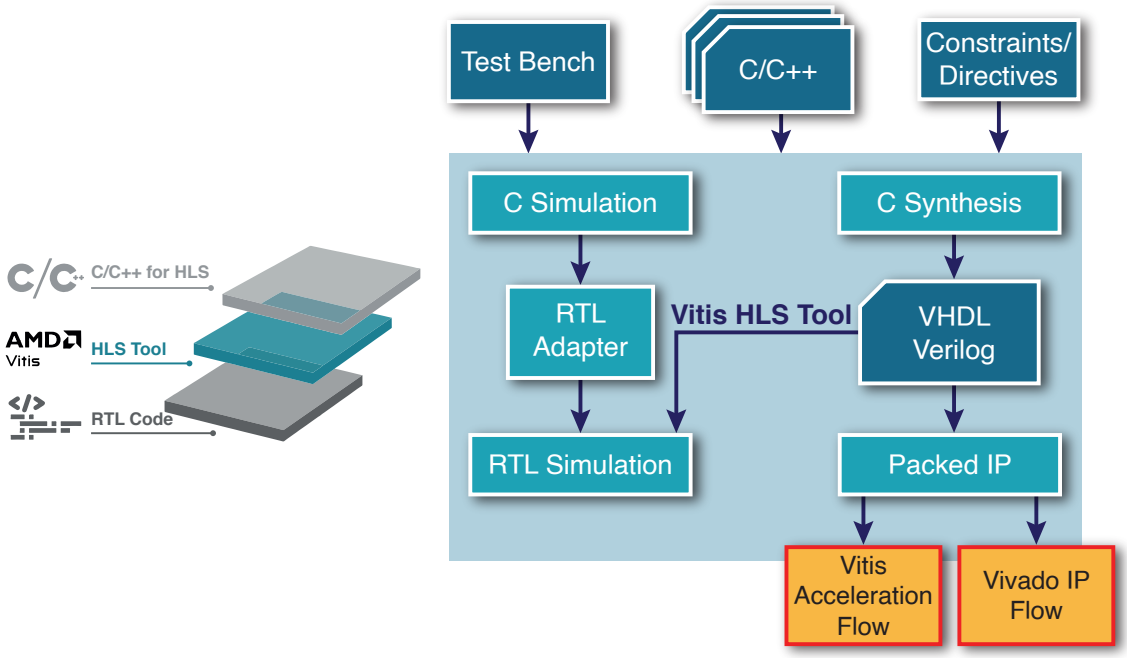


Figure 2.13: AMD Xilinx High Level Synthesis (HLS) tool chain diagram showing how C/C++ can be translated and simulated to Register Transfer Level (RTL) for FPGA design.

2.6.3 Memory

As for PS part, memory in FPGA exists in various types varying in size, properties and availability, allowing to suit best the needs of the design in throughput, memory capacity or spatial placement. The main memory elements in AMD Xilinx FPGA are: distributed RAM, BRAM and URAM as depicted in Figure 2.14. The availability of the different RAM as well as their operating frequencies depend on the architecture of the targets, so as for example URAM are only available in UltraScale+™ devices (see Table 2.1). At the hardware level, memories introduce the notion of ports that gives read and/or write access to a memory address.

Distributed RAM. Distributed RAM corresponds to combinatory logic implemented as synchronous RAM. As it names suggests, its main feature its spatial distribution so as distributed RAM are sparse small memory blocks that take advantage of a high level of ports. Using LUTRAM, 32-bit or 64-bit single or dual ports RAM can be implemented, where function generators in SLICEM can implement 512-bit single- to quad-port distributed RAM or 64-bit octal-port distributed RAM (see Figure 2.14). The main limits of distributed RAM are their size and extra consumption of resources in case of multiple clocking.

BRAM. Block RAM (BRAM) stores up to 36 Kib of data (kibi bits ($1Ki = 2^{10} = 1024$)) and can be configured as either two independent 18 Kib RAMs or one 36 Kib RAM. Each block RAM has two write and two read ports that can be configured with independent port widths for each of those ports like for example 4Ki x 9, 2Ki x 18 or 1Ki x 36 as shown in Figure 2.14. The configuration is nonetheless constrained by the operating mode of the BRAM. The benefit of the BRAM over the distributed RAM is its size and its handling of multiple clocking. Even if they are less numerous compared to distributed RAM, they feature a good spatial distribution all over the chip.

URAM. Ultra RAM (URAM) is a single-clocked, two port memory available in UltraScale+™ devices. URAM are large RAM usually available in a fewer quantity compared to BRAM but

that shows a capacity of 8 36 Kib-BRAM (see Figure 2.14). The URAM is placed in the device to allow cascading URAM column for the entire height of the device, thus forming a significantly large memory. While URAM provides significantly large memories, they also show constraints such as fixed width, single-clock only and the inability of file initialization.

Others. Other components in the FPGA can be used to memorize data in smaller sized buffer, this is notable the case of Look Up Table (LUT) or Flip-Flop (FF). FF (or registers) are synchronous elements allowing the storage of one bit available in a large quantity, allowing the use of multiple FF to memorize a data. FF are widely to buffer signals in order to maximize the operating frequency of a system as well as synchronizing signal in a pipeline architecture.

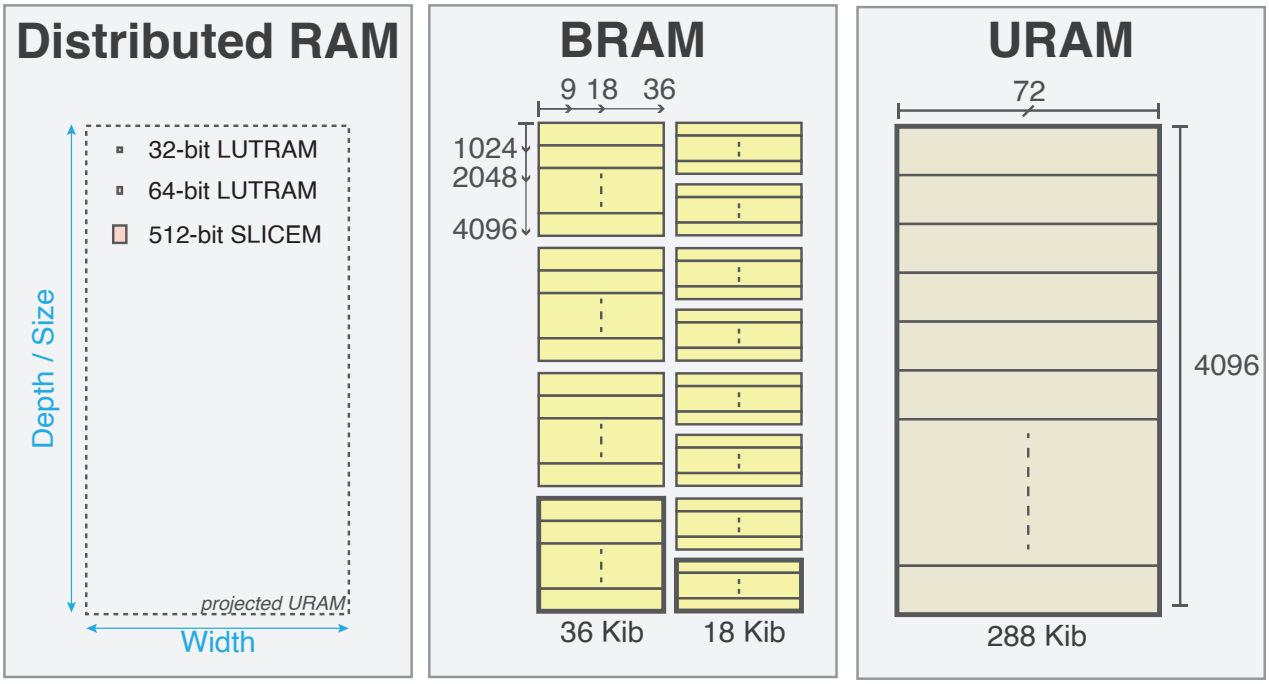


Figure 2.14: Main memory elements of Zynq UltraScale+ MPSoC PL architecture. Distributed RAM is combinatory logic implemented as synchronous RAM, BRAM are memory blocks of configurable width and URAM memory blocks of fixed width. Kib stands for kibi bits ($1Ki = 2^{10} = 1024$)

.

2.6.4 Arithmetical and mathematical operations

Due to the low-level hardware nature of FPGAs, the implementation of mathematical operations is different from software-based calculations on conventional processors that execute operations through sequences of instructions.

FPGAs directly generates a custom digital circuits to perform each computation so as this hardware-centric approach provides high efficiency gains through true parallel computation.

However, it can allow limit or complicate the implementation of mathematical operations, especially for operations on data with high level of abstraction like floating-point.

A crucial component for the implementations of mathematical operation on FPGA is the Digital Signal Processor (DSP). DSP is specifically designed and optimized to carry out performant computation like multiplication or addition. Its architecture allows running computations at high clock frequencies and its placement on the FPGA close to the BRAM allows routing and interconnecting them efficiently with the other elements of the system (see Figure 2.15).

It allows placed so as DSP can be cascaded implement large pipelined operations. In the SOM K26, the DSP are DSP48E2 that output the result P on 48 bits and can accept up to 4 inputs: A on 30 bits, B on 18 bits, C on 48 bits and D on 27 bits as shown in Figure 2.15. Examples of operations that can be performed by the DSP using the IP core include $A \times B + C$, $(A + D) \times (A + D) + C + C$ or $A \times B + P - C$.

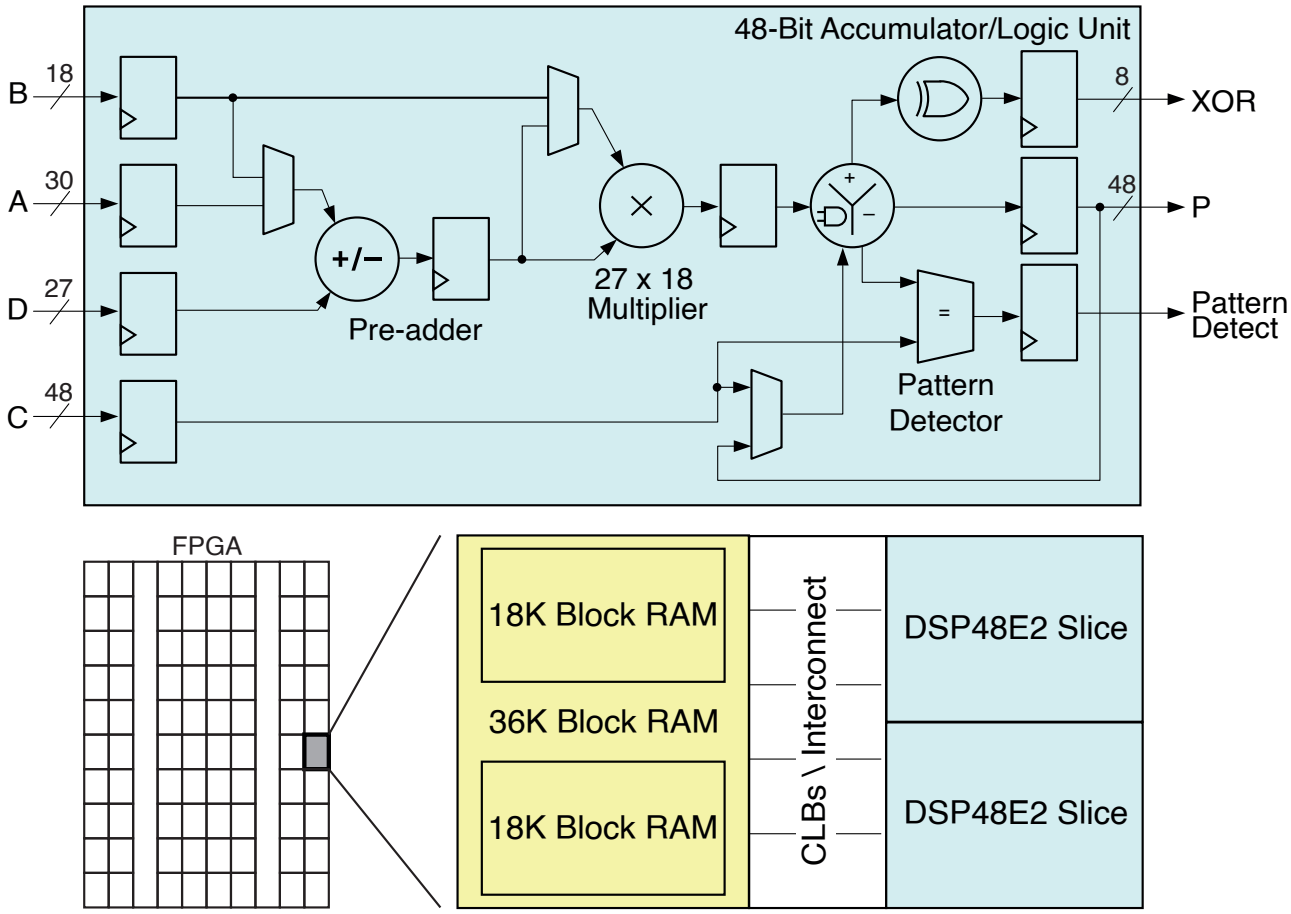


Figure 2.15: Basic DSP48E2 functionality and DSP tile interconnect in columns based from the documentation [Xilinx, 2021]. DSP are placed closed to the BRAM to allow efficient routing that ensures functioning at high clock frequencies.

Addition (add) and Subtraction (sub). Additions (add) and subtractions (sub) of integers, including fixed-point coding as its coding is essentially the same, is a simple operation in FPGA. In can efficiently be implemented with combinatory logic to perform like operations in one clock cycle, but they can also be implemented using DSP. The implementation of these operations for floating-point coding (fadd and fsub) shows considerably higher implementation cost and latency, but can still be implemented with relative ease.

Multiplication (mul). Multiplications of integers or fixed-point numbers (mul) can be performed easily and efficiently by DSP, striking a balance between latency and operating frequency. Adding more registers increases latency while also raising the operating frequency. Multiplication of floating-point numbers (fmul) is similarly performed by DSP with a slightly higher latency and implementation cost. A common optimization to replace multiplications in FPGA is to replace the multiplication of power of two by left shifts.

Divisions (div). Division of integers or fixed-point numbers (div) is a rather difficult oper-

ation to implement that requires algorithms such as Restoring Division algorithm or multiplication by the inverse. A compromise can usually be found between the hardware usage and the latency. The unbounded nature of division that can lead numbers to tend towards infinity can also be challenging to handle with integers. On the contrary, divisions of floating-point numbers can be implemented in fabric for a very low implementation cost and a fair latency. As for multiplications, a common optimization relies on a shift to the right to replace divisions by power of two.

Others. The intricacy of implementing other complex mathematical functions varies significantly based on the specific functions involved. While some IP cores or libraries may exist to implement the functions, some very specific operations require developing custom hardware or finding optimizations that will allow the use of the elementary operations available. A common example is the implementation of trigonometric functions by implementing a CORDIC (COordinate Rotation DIgital Computer) as performed by the team in [Khoyratee et al., 2019]. An approach used in the system also relies on the use of pre-computed calculations stored in RAM to implement complex mathematical functions.

2.7 Summary

In this section, we explained the choice of the digital platform selection for the development of a real-time biomimetic SNN development on a SoC FPGA. It granted knowledge of the already existing system and technological landscape that influenced and supported the choice of the targets selected. Additionally, it introduced the communication protocols and interfaces inherent to our system that are essential in the design of a system that aims to interface with biology. It also presented and detailed the different elements and characteristics of the SOM K26 with particular attention given to its two different parts PS and PL. To conclude, this chapter furnished the essential knowledge necessary to grasp a good understanding of the software and hardware design development within this specific target context.

3 Toward a flexible real-time biomimetic SNN on SOM K26

3.1 Introduction

With a deeper understanding of the artificial neural networks and the platform used for its implementation, the focus shifts on the design of the system.

Given that the system developed is intended to be used by biologists for either real-time emulation of biophysically detailed networks or hybridization purposes, it is crucial to consider key features such as flexibility and user-friendliness. In this way, non-specialists users could easily use and tune the system to suit their needs and adapt to their experimental setup.

Another important aspect of the system lies its capability to interface with other systems or components so as it must be able to interface with most standards of biophysical interfaces.

The system also requires performances to be capable of emulating a satisfying number of neurons and synapses to create a network, while ensuring its real-time behavior.

Considering the target selected, an efficient design will then involve the use of different level of abstraction, translated by different programming languages, to provide the best compromise between performances, ease of use and flexibility.

This chapter will explore the different part of the design and explain the design methods applied, starting with the hardware component operating in the PL part of the SOM. Then, the different layers of software developed responsible for the user interface, monitoring and communication will be explained. Finally, the performances of the system will be presented.

3.2 Overview

The system developed was named BiœmuS standing for "**BIO**mimetic **EMU**lation **S**ingle compartment". It corresponds to the design that is capable of running up to 1,024 single compartment neurons fully connected, supporting a total of 2^{20} synapses. It includes on-board monitoring and offers versatile external communication options such as Ethernet or WiFi.

A prototype version sharing the same base and integrating multicompartmental neurons was also developed as BiœmuM for "**BIO**mimetic **EMU**lation **M**ulticompartmental".

The system is developed using 3 different languages that corresponds to 3 distinct parts. Python language is used for the configuration scripts and monitoring to provide user-friendly and rapid-prototyping as it is aimed to be used by non-specialists. The C++ language is used to develop the application responsible for setup and control running on the SOM in the PS part to provide better performances and proximity with hardware. VHDL was used to describe the hardware circuit in the PL part of the SOM that implements the calculation core of the neural network. Some C++ description was used to generate the HLS IP used in the calculation core. The Figure 3.1 illustrates the different parts of the system and indicates their hardware or software nature for a configuration and monitoring on an external computer. The configuration of the network in Python can also be executed on the SOM thanks to the Canonical Ubuntu.

The neurons constituting the SNN are modeled with high biological plausibility using the Hodgkin-Huxley (HH) paradigm and a current mimicking synaptic noise to reproduce spontaneous activity. Neurons are connected using biomimetic synapses mimicking AMPA, NMDA, $GABA_A$ and $GABA_B$ receptors to allow fast and slow synaptic excitation or inhibition. All parameters of the HH model, synaptic noise and synapses parameters are configured from Python scripts.

Alternative prototype versions running on different target notably include a reduced version on CMOD A7, an early version on Zybo Z7-20 that implements USB communication and a multicompartmental calculation core in development that works in specific context on SOM K26.

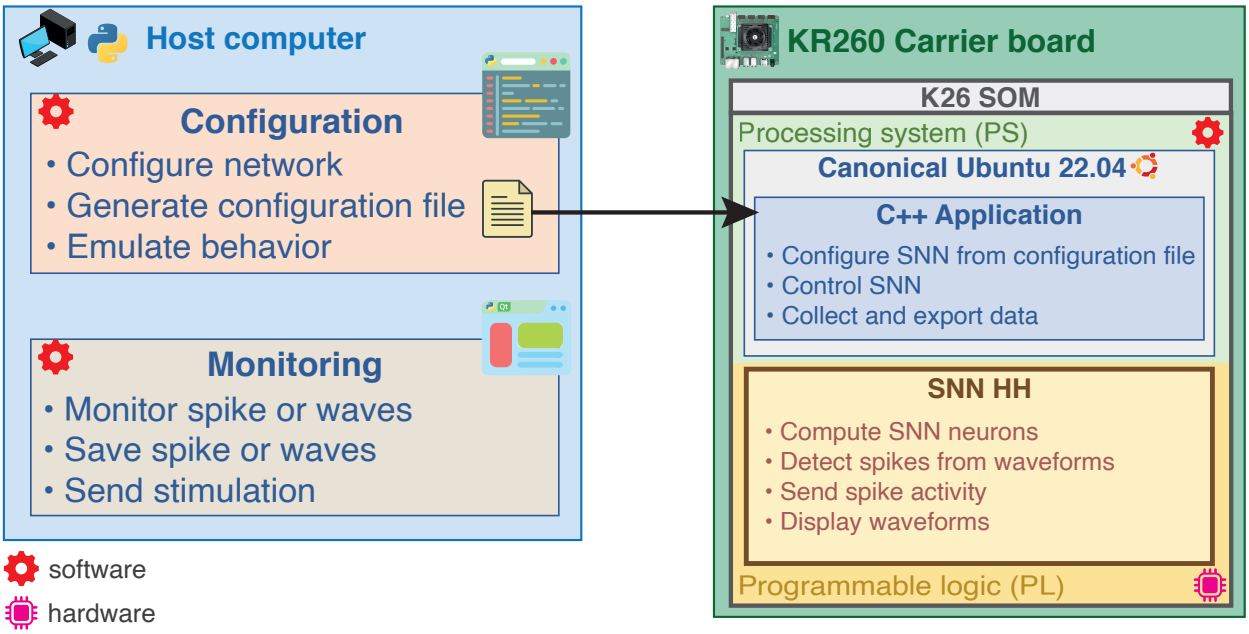


Figure 3.1: Block diagram of the global architecture of BiœmuS. The nature of each part of the system (software or hardware design) is identified by red and pink symbols. The on-board configuration and monitoring are also available but not displayed on the figure.

3.3 Hardware: Computation Core

This section will detail the hardware design corresponding to the computation part of the system. The computation core corresponds to the main part responsible for the emulation of the neurons. It includes the calculation of ion channels states, ion currents, noise and stimulation currents as well as synaptic current.

The hardware computation core for single compartment neurons in its latest version allows emulating 1,024 fully connected neurons for a total of 2^{20} synapses. The prototype computation core for multicompartmental neurons allows the computation of 16 neurons with 64 compartments without synapses.

The neurons composing the network are modeled with high biological plausibility using the HH paradigm [Hodgkin and Huxley, 1990] in the Pospichil model [Pospischil et al., 2008] implementing 6 conductance-based currents. An injected current mimicking synaptic noise following an Ornstein–Uhlenbeck process [Destexhe et al., 2001, Grassia et al., 2016] reproduces spontaneous activity by triggering action potentials on a random basis.

The computation core is clocked at 400 MHz that represents 80% of the maximum operating clocking frequency of the components used (URAM 500 MHz).
The hardware design also implements mixed data coding using both floating-point and fixed-point to provide a good balance between latency and resource-usage that directly correlates to performances.

This section presents the hardware design of the system that describe the calculation core implementing neurons and synapses to create a network. The Figure 3.2 shows the architecture of the calculation core running on the FPGA part that will be detailed in this section.

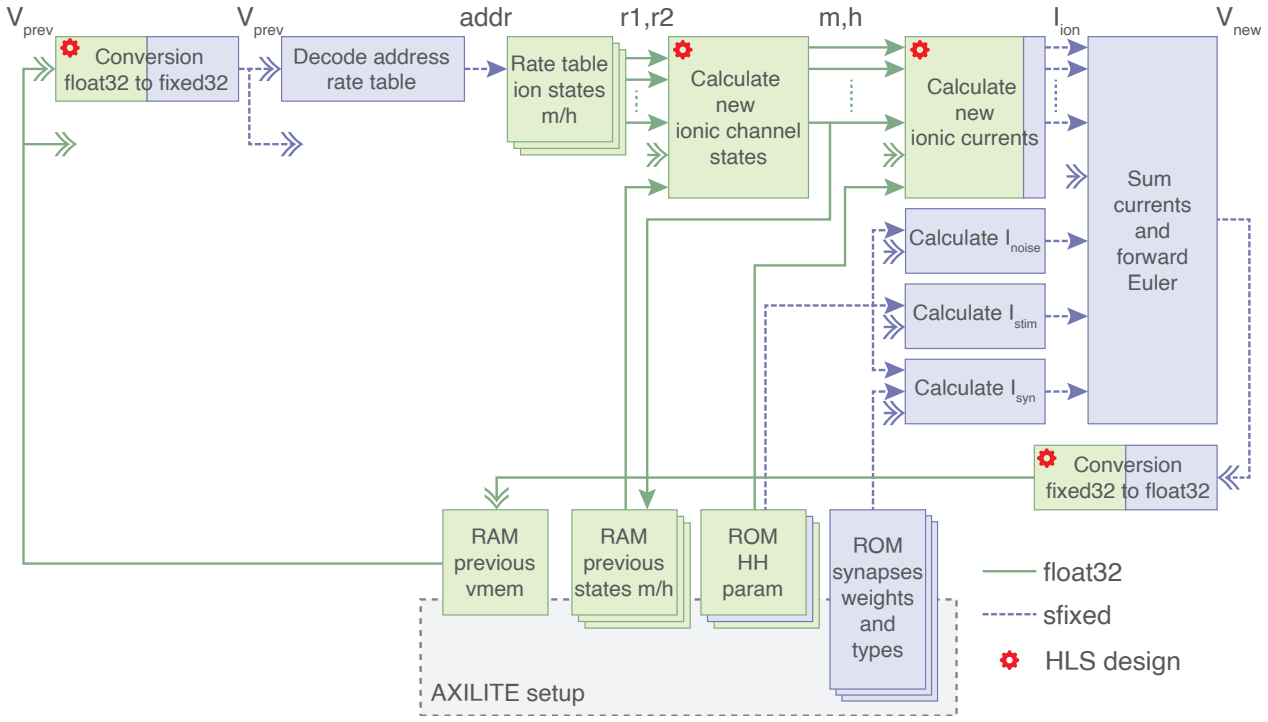


Figure 3.2: Block diagram of the calculation core architecture for single compartment neurons (BiœmuS). Green blocks and lines correspond to modules operating on fixed-point coding and blue blocks and lines to floating-point.

3.3.1 Ion channels states

The ion channels states in the HH model correspond to the variable responsible for the activation and inactivation of the ion channel ruled by equations of the form of Equation 1.7 or Equation 1.8 previously introduced in chapter 1 (section 1.5.4). The ion channel states are sigmoid functions evolving between 0 and 1 and varying depending on the ion channels.

The arithmetical functions ruling the ion channel states usually involve division and exponential, two operations that does not suit well the FPGA architecture.

The approach previously used in the team relied on the use of a CORDIC (COordinate Rotation DIgital Computer) and fitted equations to describe all ion channels states dynamics as hyperbolic tangents or hyperbolic cosinus [Khoyratee et al., 2019].

However, this approach shows limitations in terms of the equations that can be implemented so as it enforces to perform a fitting of the equations and limits the dynamics that can be encoded.

Another approach explored in this system was the use of pre-computed tables stored in RAM to reduce the calculations of the ionic channel states as proposed in [Hines, 1984]. The calculation of the ion channel states based on a restated equation of the forward Euler solving then corresponds to a simpler equation that relies on two pre-computed tables stored in RAM that corresponds as equated in Equation 3.1.

$$x_{n+1} = r_1(V_n) \times x_n + r_2(V_n) \tag{3.1}$$

where, x_{n+1} and x_n are respectively the new and current value of the ionic channel states, V_n is the membrane voltage at previous time step, r_1 and r_2 are the ion rate tables decoded from membrane voltage.

More concretely, this method applied to the Equation 1.7 and Equation 1.8 gives respectively the Equation 3.2 and Equation 3.3.

$$\begin{aligned} r_1(V) &= 1 - dt \times (\alpha_x(V) + \beta_x(V)) \\ r_2(V) &= dt \times \alpha_x(V) \end{aligned} \tag{3.2}$$

$$\begin{aligned} r_1(V) &= 1 - \frac{dt}{tau_x(V)} \\ r_2(V) &= \frac{dt \times x_\infty(V)}{tau_x(V)} \end{aligned} \tag{3.3}$$

where, r_1 and r_2 are the pre-computed rate table for ionic channel states decoded from the membrane voltage, dt the time step in ms, $tau_x(V)$, x_∞, α_x and β_x the equation of the ionic channel state depending on the formalism used.

The Equation 3.1 can be easily implemented on FPGA using DSP connected directly to memories, thus providing good performances in terms of latency, operating frequency and resources usage.

Nonetheless, this method introduces a balance between the implementation cost and accuracy through the size of the pre-computed RAM, larger RAM would provide better accuracy but consumes more memory. The Figure 3.3 demonstrates that principle by comparing the membrane potentials with and without the method for 4 types of neurons FS, RS, IB and LTS depending on 3 sizes of pre-computed RAMs.

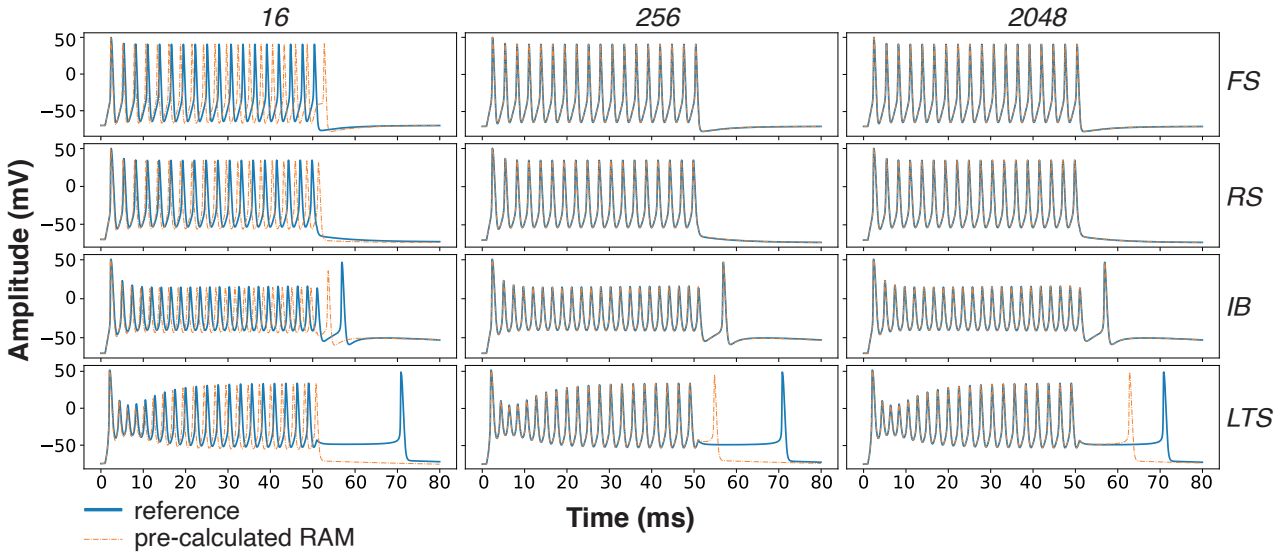


Figure 3.3: Comparison of the balance between the size of the pre-computed RAM (16, 256 and 2048) and its accuracy on the emulation of FS, RS, IB and LTS neurons in response to a 50 ms stimulation step. The simulation was performed in software with for sole difference the use of the pre-computed RAM.

It points out that for some type of neurons that have fast dynamics like FS or RS neurons, small tables are sufficient to obtain satisfying fitting. On the other hand, slow dynamics like in LTS neuron require larger tables to fit the original behavior.

Depending on the size of the tables, different memory types can be used. For 256-element table, the implementation in distributed RAM is the most suitable as it shows good performances for a little resource usage. The use of BRAM for small tables is not optimized and would constitute a waste of resources since the table would not be completely filled. An optimized use of the BRAM would be to have 1024 or 2048 points so as the RAM would be fully filled for BRAM 18Ki and 36Ki.

As the dynamics of the ion channels may vary greatly as for example in LTS where the Sodium current is significantly stronger than the Calcium current, the data encoding required high accuracy. Thus, all ion channel states were encoded using single floating-point (32 bits). Fixed-point coding would have required signals as large as 32 bits, thus representing the same memory footprint as single floating-point and relatively close implementation cost for multiplications and additions performed by DSP. The aim of providing a flexible platform also pushed the use of floating-point that allow finer tuning of the parameters of the system without limitations in ranges imposed by fixed-point representation.

The design of the module that perform the calculation of Equation 3.1 was designed using Vitis HLS (see Figure 3.2) to develop in a faster way the implementation of floating-point coding that is less straightforward in VHDL with IP cores.

The previous values of the ion channel states were stored in BRAM to allow a good scaling in terms of number of neurons (1152 neurons per 36 Kib BRAM) and performances with DSP calculations.

The setup of the pre-computed memory by the software part of the system through AXI-Lite will be further detailed in the setup part in this section.

3.3.2 Single compartment neurons

The equation of the HH neuron model were slightly modified for the implementation on the FPGA to save resources and better takign advantage of the FPGA architecture. As equated in Equation 1.16 previously detailed in chapter 1 (section 1.5.4), the sum of the ion currents is multiplied by $\frac{\Delta t}{C_m}$. In order to save memory resources, all conductances were pre-multiplied by that coefficient, thus saving the use of a memory to store C_m for each neuron, a division and a multiplication by dt.

The parameters of the neuron model are summarized in the Table 3.1 as well as the preset values for FS, RS, IB and LTS neuron types. The parameters of the model for each neuron are stored in BRAMs so as each parameter is allocated 1 BRAM, thus allowing up to 1152 neurons per parameter. The setup of the parameters is done the software part through AXI-Lite and will be further detailed in another section.

The data encoding of the neuron computation block features both fixed-point and single floating-point coding to obtain a good compromise between performances and accuracy.

The computations of ion currents are performed using single floating-point 32 bits as the operations are essentially multiplications with only one addition/subtraction efficiently performed by DSP and some combinatory logic. The other currents that require less accuracy such as stimulation, noise and synapses are performed using fixed-point coding (mostly 18 and 25 bits) as shown in Figure 3.2.

The addition using large fixed-point signals can show better accuracy than floating-point and do show significantly better performances in terms of implementation cost and latency. Thus, the all currents are transcoded from single floating-point 32 bits to fixed-point 32 bits,

then summed to obtain the new membrane potential (see Figure 3.2). The membrane voltage is then transcoded back to floating-point to be stored in the RAM for previous state.

While most of the dynamics can be encoded using less resources with large fixed-point coding (32 bits), the memory footprint is similar to single precision floating-point (32 bits). Following the aim of flexibility of the system, floating-point coding shown acceptable increase of latency and resources usage for the flexibility gain brought to the system by floating-point coding.

Similarly to ion channels states, the computation of the ion currents computation were performed using Vitis HLS, thus providing an extensive gain of development time and genericity as a new current can be easily added to the module from C/C++ language.

Table 3.1: Parameters of the Hodgkin-Huxley model for the 4 preset types of neuron tunable from the Python scripts.

Parameter	FS	RS	IB	LTS	Unit
g_{Na}	0.05	0.05	0.05	0.05	S/cm^2
g_K	0.01	0.005	0.005	0.005	S/cm^2
g_M	0.0	7×10^{-5}	3×10^{-5}	3×10^{-5}	S/cm^2
g_L	0.0	0.0	1×10^{-4}	0.0	S/cm^2
g_T	0.0	0.0	0.0	4×10^{-4}	S/cm^2
g_{Leak}	0.00015	0.0001	1×10^{-5}	1×10^{-5}	S/cm^2
E_{Na}	50.0	50.0	50.0	50.0	mV
E_K	-100.0	-100.0	-90.0	-100.0	mV
E_{Ca}	0.0	0.0	120.0	120.0	mV
E_{Leak}	-70.0	-70.0	-70.0	-75.0	mV
v_{init}	-70.0	-70.0	-70.0	-75.0	mV
μ_{noise}	0.048	0.042	0.042	0.042	1
θ_{noise}	8.0	8.0	8.0	8.0	1
σ_{noise}	0.11	0.09	0.09	0.09	1
I_{stim}	0.03	0.03	0.03	0.03	mA/cm^2
$area$	$(67 \times 10^{-4})^2$	$(96 \times 10^{-4})^2$	$(96 \times 10^{-4})^2$	$(96 \times 10^{-4})^2$	cm^2
c_{mem}	1.0	1.0	1.0	1.0	$\mu F/cm^2$

The time step selected for the computation of the neuron was 2^{-5} ms, as in [Khoyratee et al., 2019], to ensure a satisfying stability and accuracy for the system solving with Forward Euler solving. The Figure 3.4 shows the membrane potential corresponding to the implementation of FS, RS, IB and LTS neurons using the parameters of Table 3.1 and 2048-element ion channel states tables in response to a 10 ms stimulation.

The Figure 3.4 shows a good fitting for each type of neuron in response of a short stimulation time. However, the LTS neuron can show less accurate fitting for longer stimulation as highlighted by Figure 3.3 in the case of a 50 ms stimulation in software.

The computation latency of the calculation core for only neurons without synapses is of 96 clock cycles. At the current clock of 400 MHz and time step of 31.25 µs (2^{-5} ms), the calculation core would allow the computation of at least 11,000 independent neurons.

The implementation of the neurons mainly uses DSP, LUT and BRAM. The detailed re-

source utilization associated with the implementation of neurons in the computation core will be presented further.

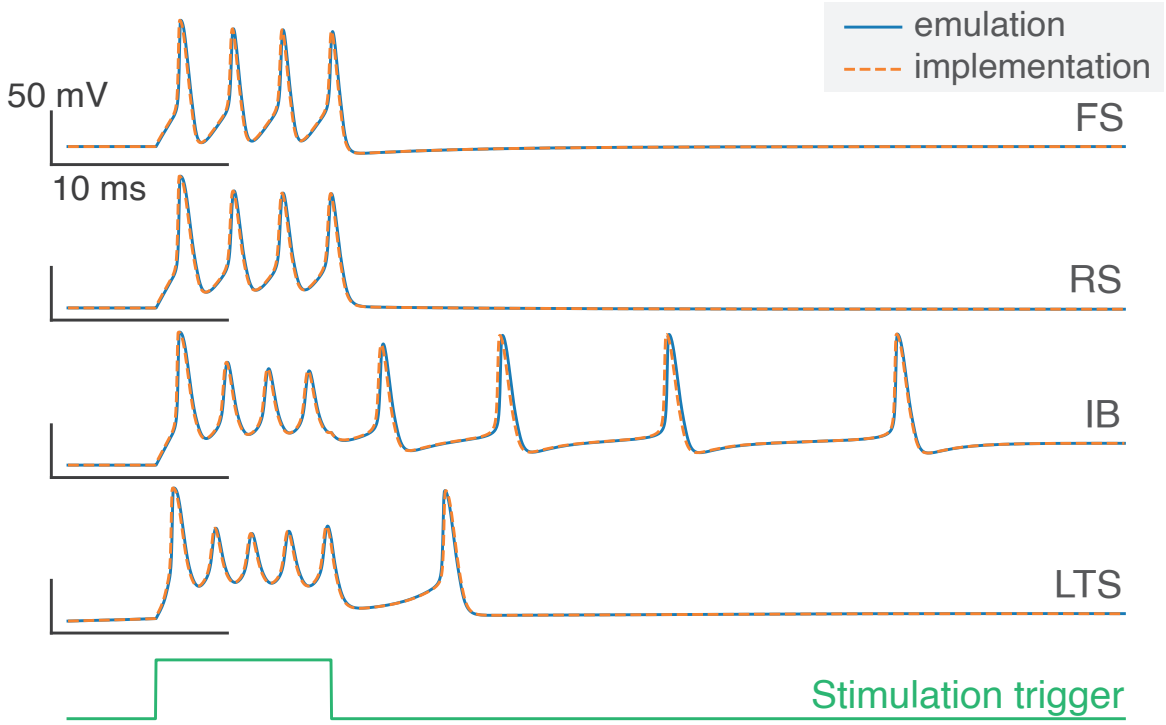


Figure 3.4: Comparison of membrane voltage response in implementation to a 10 ms current injection for Fast Spiking (FS), Regular Spiking (RS), Intrinsic Bursting (IB) and Low-Threshold Spiking (LTS) neurons. Waveforms were captured using the on-board saving of the system. Emulation corresponds to the software emulation of the system through the Python scripts and implementation to the waveforms monitored on the system.

3.3.3 Multicompartmental neurons

The computation core emulating multicompartmental neurons in BiœmuM presents a different architecture mostly explained by the use of a different solver. As presented in the chapter 1 section 1.6.3, the Crank-Nicholson solver is used thus equating the evolution of the membrane potential in a mainly tridiagonal system of equations.

As explained in [Hines, 1984], using matrix inversion to solve the system is a resource-intensive approach so as a wise numbering of the compartments would allow efficient solving using the Hines algorithm.

However, this algorithm was tailored for CPU architecture so as variant of the algorithm suiting better the GPU architecture are used [Valero-Lara et al., 2018]. This is notably the case of a GPU-based simulator Arbor developed by the Human Brain Project community [Akar et al., 2023].

The algorithm uses a parent node vector p so as the matrix can simply be stored using two vectors corresponding to the main diagonal D and upper diagonal U. Branching points are then reconstructed thanks to the parent node vector. The Algorithm 2 generates the main diagonal D and the Algorithm 1 solves the matrix.

The computation core for multicompartmental neurons thus requires to update the ion currents computation to output two separate coefficients D and B. It also requires two computation blocks that perform the backward and forward sweep of the Algorithm 1.

The coefficients for each segment are then stored in one dual-port RAM (lower addresses is D and upper addresses are B) with one BRAM per neuron. These BRAM act as buffer memory for the operations of the forward and backward sweep by loading and storing the values of the segments at each iteration until the matrix is completely solved.

Algorithm 1 Hines algorithm used in Arbor simulator

```
1:  function SOLVE(d, u, p, b, nseg)
    // Backward sweep
2:      for i ← nseg − 1 to 0 step −1 do
3:          factor ← u[i]/d[i]
4:          d[p[i]] ← d[p[i]] − factor · u[i]
5:          b[p[i]] ← b[p[i]] − factor · b[i]
6:      end for
    // Forward sweep
7:      b[0] ← b[0]/d[0]
8:      for i ← 1 to nseg step 1 do
9:          b[i] ← b[i] − u[i] · b[p[i]]
10:         b[i] ← b[i]/d[i]
11:     end for
12:     return b
13: end function
```

Algorithm 2 Initialize format main diagonal

```
1: function FORMATD(d, u, p, nseg)
2:     for s ← 0 to nseg − 1 do
3:         d[s] ← d[s] − u[s]
4:         if p[s] ≠ −1 then
5:             d[p[s]] ← d[p[s]] − u[s]
6:         end if
7:     end for
8: end function
```

The updates of the computation core for the handling of multicompartmental neurons are summarized in Figure 3.5. While the main changes concern the computation core, other miscellaneous changes are required in the architecture to allow multicompartmental modeling.

As the solving algorithm of the matrix includes sequential divisions and multiplications, the stability of the solving requires high accuracy that is better translated by floating-point. Indeed, the coefficients greatly vary with the geometrical dimensions of the neurons that may create larger order of magnitude delicate to handle with fixed-point.

A compromise can be found in the design to either favor a scaling in neurons or segments. The current implementation corresponds to a naive approach where each segment corresponds to a backward and forward cell that allow a better scaling in neurons as all neuron solving operate concurrently. However, the resource utilization will greatly limit the number of segments that can be implemented.

Another approach would rely on the use of only one backward cell and one forward cell shared between neurons using a FIFO that memorizes the operations to be computed in a queue. This approach would allow minimal resource utilization thus a good scaling in segments but might limit the number of neurons and segments because of the computation time. Indeed,

as the latency of the backward and forward cell is not negligible, a bottleneck would be formed by the number of computations one cell can accept in the pipeline.

The best solution would be a compromise of these two approach by implementing a limited number of backward and forward cells shared between neurons. Hence, a balance between the resource utilization and performances can be found. The Figure 3.5 presents the architecture corresponding to the architecture described.

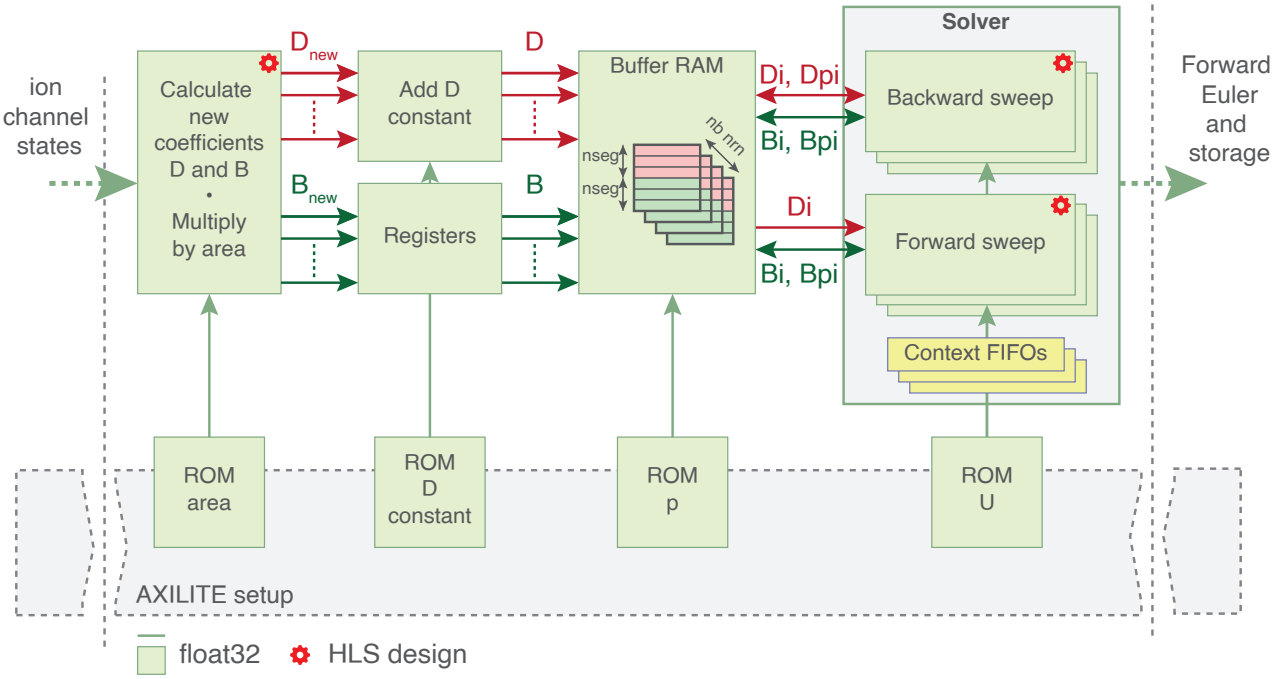


Figure 3.5: Block diagram of the difference for multicompartmental computation core. The ion currents are split into two coefficients D and B. Dual-port buffer RAM for D and B of each neuron loads and stores with the forward and backward sweep cells.

Similarly to the computation core for single compartment neurons, a software emulation model in Python was developed to ease the FPGA development by providing an emulation of the hardware architecture to predict the implemented behavior. The Figure 3.6 presents a comparison with the NEURON software [Carnevale and Hines, 2006] targeting the emulation of a motor neuron at early stage that will be further detailed in Chapter 4.

In Figure 3.6, the soma of a motor neuron including only sodium, potassium and leak currents is modeled using 5 segments of identical length, diameter and properties. A stimulation current is inserted at 1 ms in the first segment for 10 ms and propagated in the others segments of the section, thus validating model thanks to the propagation of the stimulation to adjacent segments. As the segments are of identical and relatively small length and properties, the signal propagated is identical and almost instantly propagated.

The differences observed between the emulation and the NEURON software are explained by the difference of solver that is CVODE for NEURON and Crank-Nicholson for the software emulation, but mostly by the hardware architecture constraints in terms of data coding and operations. Indeed, the hardware is operating on mixed fixed-point / 32-bit floating-point operated on hardware by a FPGA instead of 64-bit floating-point on software with a CPU for NEURON.

In conclusion, the computation core for multicompartmental model features an architecture of higher complexity and resource utilization than the single compartment core. It also introduces a compromise in the architecture designed that could either favor the scaling in neurons

or in segments. In this manuscript, the prototyped core designed is promoting the scaling in segments rather than neurons as the emphasized is put on the morphology of neurons so as the more compartments, the better.

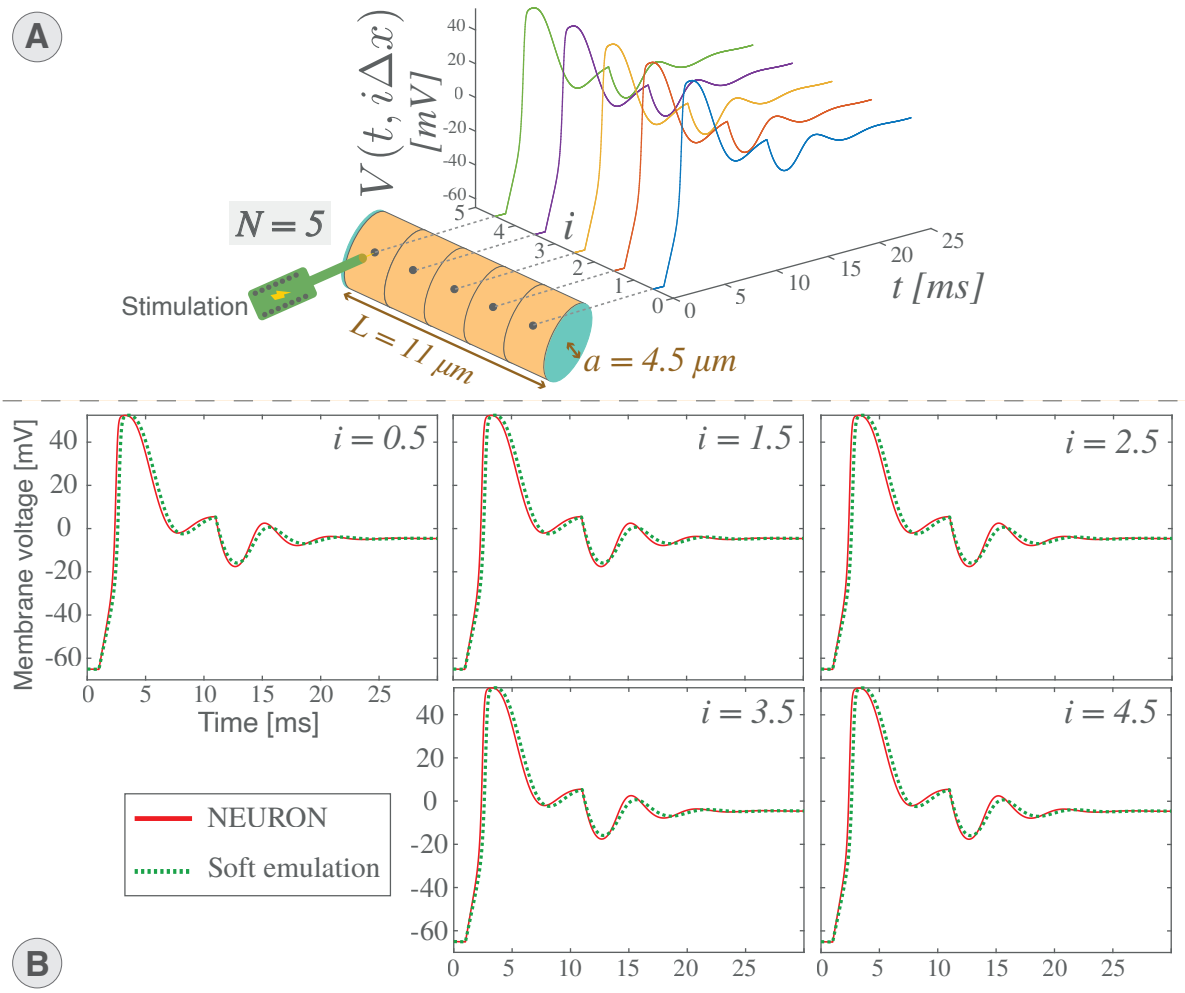


Figure 3.6: Software emulation of the hardware architecture for the soma of a motor neuron section in response to a 10 ms stimulation in the first segment. **(A)** Software emulation of the hardware architecture for a section of 5 segments. **(B)** Comparison of the software emulation with similar emulation using NEURON software.

3.3.4 Synapses

The synapses implemented follows the Destexhe model [Destexhe et al., 1998] that models four receptors AMPAR, NMDAR, $GABA_AR$ and $GABA_BR$. Similarly to the ion channel states, the sigmoid functions in synapses such as T_{syn} or B_{syn} were pre-computed and stored in memory similarly to ion channel states. The calculation then corresponds to Equation 3.4 and Equation 3.5.

$$T_{syn}(V_{pre}) = \frac{T_{max}}{1 + e^{\frac{-(V_{pre} - V_p)}{K_p}}} = T_{rate}(V_{pre}) \tag{3.4}$$

where, T_{rate} is the pre-computed memory decoded from the value of the previous membrane voltage V_{pre} and K_p (5 mV), V_p (2 mV) and T_{max} are constants.

$$B_{syn}(V) = \frac{1}{1 + \frac{e^{-0.062 \times V} \times Mg2+}{3.57}} = B_{rate}(V_{pre}) \tag{3.5}$$

where, B_{rate} is the pre-computed memory decoded from the value of the previous membrane voltage V_{pre}, $Mg2+$ (1 to 2 mM) is a constant.

The synaptic conductances were computed in nS to allow calculation of variables in the same order of magnitude, then the current is multiplied at the end by a coefficient p_{mul} that for unit consistency of units. This coefficient also allow to create a scaling factor of the network by assigning a stronger weights to synapses, hence corresponding to the influence of more synapses and mimicking a larger network. The Figure 3.7 shows the architecture of the calculation core for the synapses.

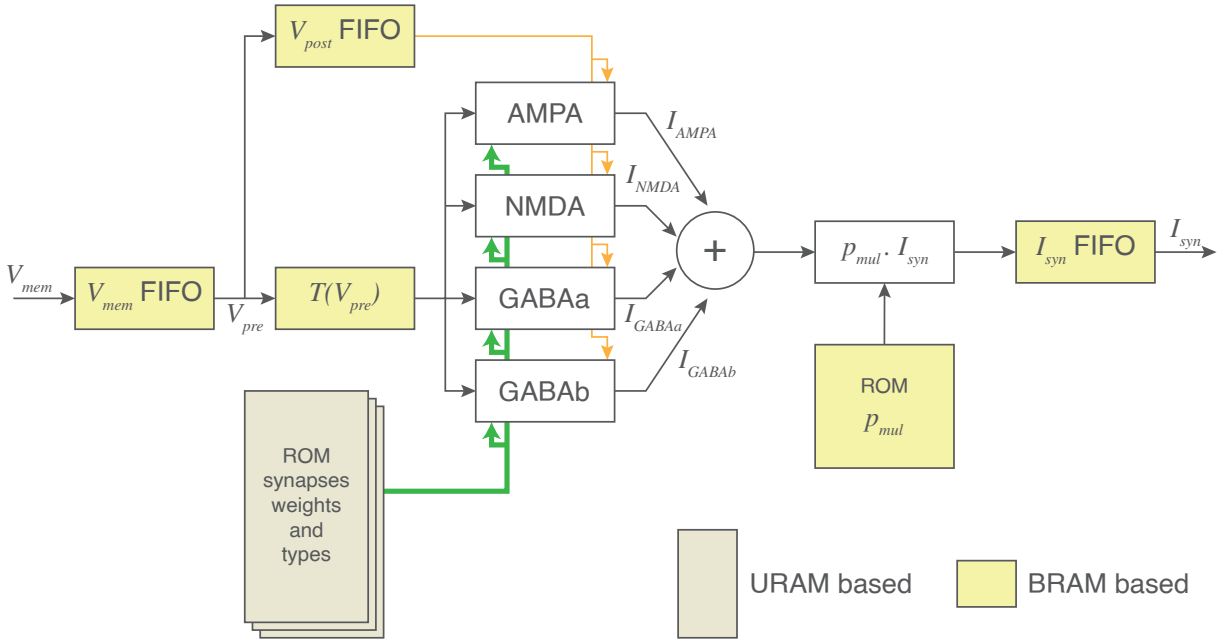


Figure 3.7: Block diagram of the synaptic current calculation module using fixed point coding. The type of memory implemented is shown for blocks integrating FIFOs, ROM or rate tables. The p_{mul} ROM corresponds to pre-multiplied coefficients including a scale factor to mimic larger network through stronger synaptic current.

Calculations in the synapse module are carried out using fixed-point coding to optimize resource utilization while maintaining accuracy by choosing signals of width corresponding to the maximum range of the DSP. The weight and types of the synapses are coded using 16 bits to simplify the memory filling and decoding, thus allocating 14 bits to weight. The current architecture stores the types and weights of the synapses in the URAM memories to allow the largest storage size.

The calculations of the synaptic currents are essentially based on a sequential multiplication and additions that can efficiently be performed by DSP, providing high operating frequency as well as limited resource usage.

In order to efficiently interface the DSP with the memory, the principle of virtual RAM was applied so as the weights were distributed over several RAMs to be interpreted differently from their hardware structure.

The Figure 3.8 illustrates the basic calculation used in the synapse module as well as how it is implemented in hardware using URAM memory.

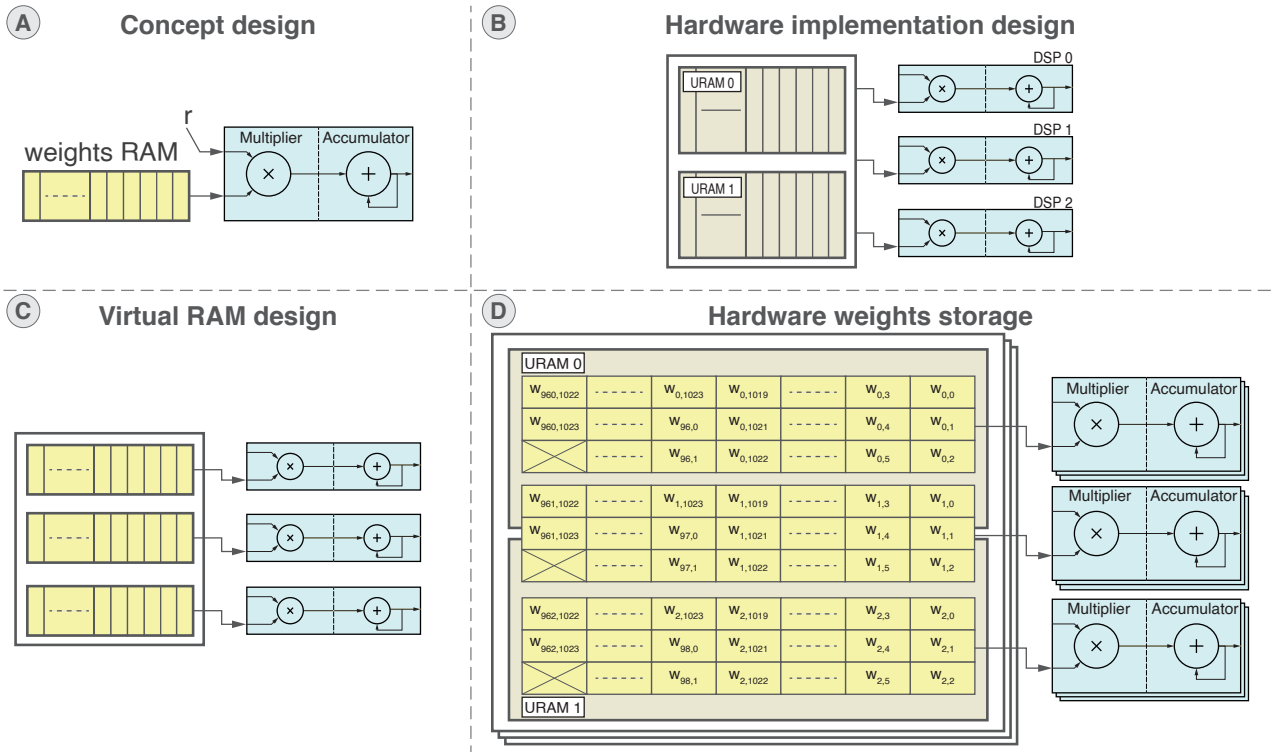


Figure 3.8: Concept and implemented block diagrams of calculations and memory organization for the synaptic current calculations. (**A**) Concept design of the synaptic current calculation where a DSP is configured in multiply-accumulate mode and connected to a RAM storing the weights. (**B**) Hardware implementation of (A) using URAM to store the synaptic weights (**C**) Virtual design of synaptic weights RAM that shows how RAM are interpreted for calculations. (**D**) Hardware implementation of (C) that presents the actual hardware storage of the weights distributed in multiple URAM.

The parameters of the synapses are shared by all the neurons so as are stored in registers setup by AXI-Lite, allowing to change the parameters of the different types of receptors of the network. The parameters of the synapses correspond to the Table 3.2 corresponding to the equations of the synapses from [Destexhe et al., 1998].

Table 3.2: Preset parameters of the synapses using Destexhe model from [Destexhe et al., 1998] tunable from the Python scripts.

Parameter	AMPAR	NMDAR	$GABA_AR$	$GABA_BR$	Unit
g	0.35	0.3	0.25	1.0	nS
E	0	0	-80	-95	mV
α	1.1×10^6	7.2×10^6	5×10^6	–	$M^{-1}sec^{-1}$
β	190	6.6	180	–	sec^{-1}
K_1	–	–	–	9×10^4	$M^{-1}sec^{-1}$
K_2	–	–	–	1.2	sec^{-1}
K_3	–	–	–	180	sec^{-1}
K_4	–	–	–	34	sec^{-1}
K_d	–	–	–	100	μM^4
n	–	–	–	4	1

The Figure 3.9 shows a validation of the behavior of each synaptic receptor using the parameters presented in 3.2.

The excitatory synapses mimicked by AMPAR and NMDAR are tested by connecting a stimulated spiking neuron connected to non spiking neurons.

The inhibitory synapses mimicked by $GABA_AR$ and $GABA_BR$ are tested by connected together stimulated neurons so as the inhibitory synapses would prevent neurons from spiking.

The Figure 3.9 validates this behavior by showing the fast and slow excitation and inhibition of synapses in implementation.

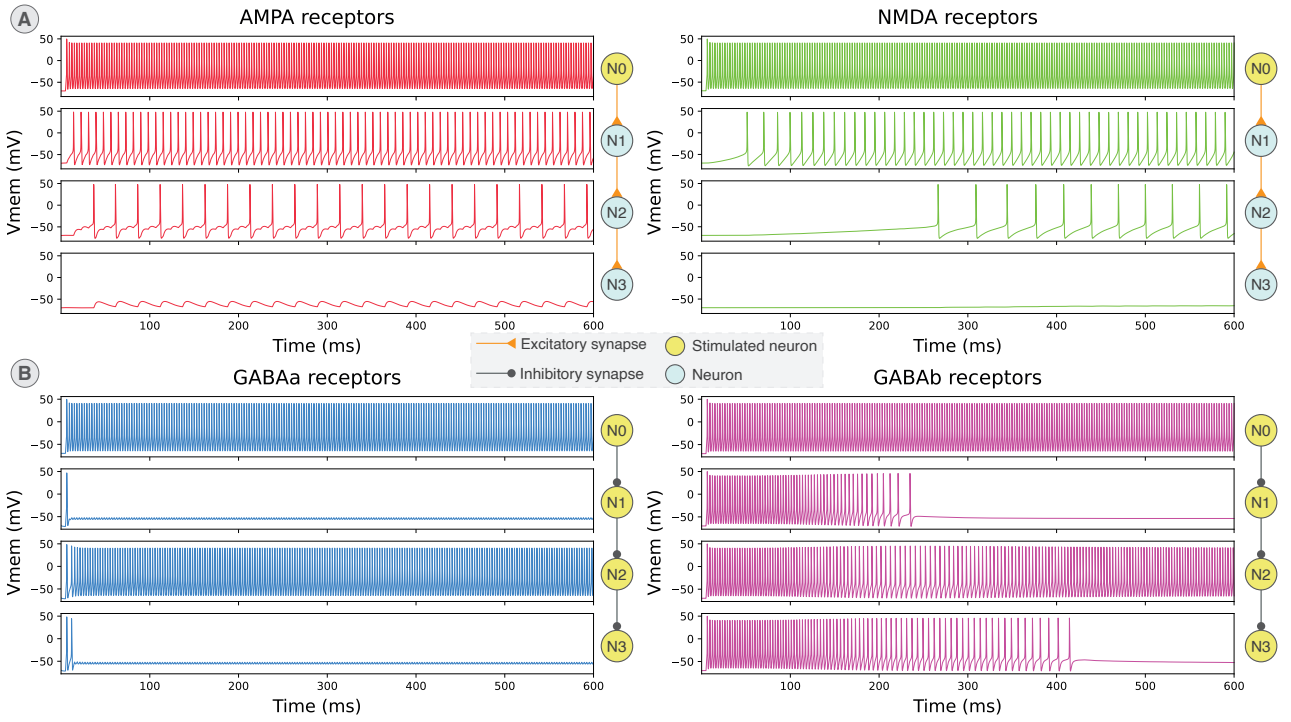


Figure 3.9: Behavior of a network of 4 identical neurons where only the type of synapses changes. The network connectivity is a chaser: neurons are interconnected with a single synapse, from N0 to N1, N1 to N2 and N2 to N3. A factor is applied to synaptic weights in order to see an excitation or an inhibition despite having only one incoming synapse on each neuron. **(A)** Only N0 is stimulated: AMPAR shows a faster response of excitation than NMDAR, synaptic weight of N2 to N3 is too small resulting in a lack of excitation of N3 thus preventing spiking. **(B)** All 4 neurons are stimulated: $GABA_AR$ shows a faster response of inhibition than $GABA_BR$.

3.3.5 Setup

The hardware design setup for BiœmuS and BiœmuM corresponds to the hardware locked properties of the system and to the dynamic setup.

The hardware locked properties correspond to settings like the clock frequency, maximum memory sizes or PMOD ports. They are initialized from the VHDL package file to allow generic design and easier maintenance and update for future developers.

The dynamic setup corresponds to the artificial model parameters, synaptic connections and monitoring parameters *via* AXI-Lite.

The AXI-Lite setup of RAMs is done using a loopback register —the writing address is written back in another register. The variables are simply set up by latching the value of the

AXI-Lite registers. Since the AXI-Lite is in a different clock domain, in most cases the parameters are set on dual-port RAMs with a write port with the same clock as the AXI-Lite. In other cases, parameters are synchronized in the other faster clock domain using double flip-flops.

The dynamic setup of the different parameters of the system is essential to the flexibility of the system by allowing online tuning. The AXI-Lite protocol is the most suited for such task and allows setting multiple registers associated to parameters. It can be simply interface with the computation core through an AXI port and shows low resource utilization. It is also well suited for the instantiation of multiple cores by adding another AXI port and setup module.

3.4 Hardware: Monitoring

The hardware design responsible for the monitoring corresponds to the components and processes that allow the monitoring of the spikes and membrane potentials waveforms of the neurons. The monitoring can be handled through three different ways operating concurrently: AXI-Stream to DMA, SPI to the DAC on PMOD DA4 and SPI to the ESP32 processor on PMOD ESP32.

This section details the hardware design that allows communication between the computation core and the different monitoring elements.

3.4.1 Spikes and waves *via* DMA

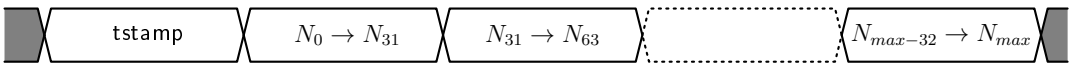
The spikes and membrane potential waveforms of the neurons can be monitored in PS using the DMA interfaced with the system using the AXI-Stream protocol and a FIFO in packet mode.

The stream of spikes and waves are stored in a FIFO in PL connected to a DMA for each stream (spikes and waves). A FIFO in packet mode starts transferring to the DMA only if its full or reached the programmable threshold. This allows to set different data collection interval for each stream by having a dynamic programmable threshold set by AXI-Lite.

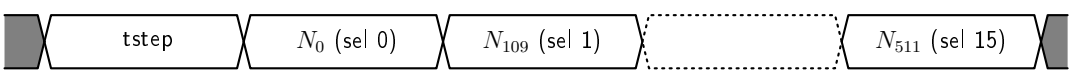
As the data collection intervals for spikes and waves are largely different in most cases (tens or hundreds of milliseconds for spikes and tens of milliseconds or less for waves), two independents DMA were implemented instead of a Multichannel DMA (MCDMA) that would struggle switching streams.

The FIFO also implements independent clocking so as the data in the clock domain of the computation core can be moved to the clock domain of the AXI protocol used for the DMA. Both spikes and waves are organized in frames starting with time stamp or time step followed by data in a stream using 32-bit signals.

Spikes frame. The spikes are compressed over a time stamp of 1 ms that is sent with each frame followed by the spike activity coded with 0 for no activity and 1 for spiking. The spiking activity of all neurons are sent in a frame.



Waves frame. The membrane potential waveforms corresponds to the values at each time step for a given number of neurons. The maximum number of neuron selected is fixed in hardware to 16 but could be increased to fit specific needs for a slight increase of resources used. The frame is constituted of a time step index and the membrane potential of all selected neurons coded in 32-bit floating-point.



The monitoring of spikes and waves are then running concurrently in hardware so as the data collection interval for both channels have no impact on each other at the hardware level. As for the limits of the data collection interval, the lower bound is limited to 1 time step (waves) or 1 time stamp (spikes) and the higher bound depends on the size of the FIFO. The bigger the FIFO, the higher the data collection interval.

The setup of the DMA interfaces like memory map width or stream width can also be tuned to maximize the performances. The current method uses a stream width of 32 bits that fits single floating-point coding and memory map width of 64 bits that suits better the PS handling.

3.4.2 Spikes *via* PMOD ESP32

The PMOD ESP32 features a radio with support for 802.11 b/g/n Wifi and dual-mode Bluetooth as well as Tensilica Xtensa microprocessor, usually referred to as ESP32.

This main benefits offered by this solution is flexible approach for interconnection of the system that suit well in-vivo applications where cables are a concern, while maintaining a low latency and acceptable throughput. In addition, this constitutes a reusable element to build a reduced and minimal embedded version of the system targeting a smaller programmable logic only target to create an energy-efficient solution for embedded applications.

The stream of spikes is sent to the microcontroller using SPI protocol so as the implementation cost of this solution on the PL part is very low. Thus, using multiple PMOD to monitor multiple cores would come at a very low-cost for the PL design.

The microcontroller is programmed to execute two tasks running concurrently using FreeRTOS so as a task constantly poll the SPI until it gets all the frame that is then passed to another tasks that is responsible for sending the frame through UDP Wi-Fi.

The PMOD is simply plugged on one of the PMOD header present on the KR260 board. The PMOD is fixed by hardware but can easily be modified by modifying the pins to obtain a different physical configuration of the system.

3.4.3 Waves *via* PMOD DA4

The PMOD DA4 features eight 12-bit channels allowing to output up to 8 membrane potentials that allows a quick real-time visualization of the waveforms. The ESP32 is a versatile microcontroller widely used for various applications like IoT, embedded systems and wireless communication projects thanks to its Wi-Fi and Bluetooth capabilities. The main benefit of this solution is to propose a wireless.

The maximum operating clock frequency of the PMOD DA4 is 50 MHz thus requiring a different clock domain. The stream of membrane potentials is synchronized from the faster clock domain to this slower clock domain using extension and latch of the data for each channel. Each channel selects the neuron waveform to display by comparing the neuron index to the value of set by the AXI-Lite. The hardware design makes it so as the neuron select can be changed online and be effective as long as the AXI-Lite register value is updated.

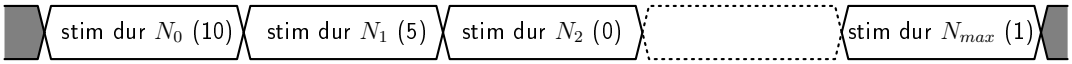
Similarly to the PMOD ESP32, the PMOD is simply plugged on one of the PMOD header present on the KR260 board fixed by hardware that can be easily modified by modifying the pins to obtain a different physical configuration of the system.

3.4.4 External stimulation

An external stimulation slot is available in hardware to trigger stimulation in each neuron for a given time interfaced with the read channel DMA also used for spikes monitoring. This allows efficient and versatile interconnection of the system with PL through the PS.

A stream of data containing the stimulation period for each neuron is sent to the PL using AXI-Stream protocol from the DMA. The stimulation period in multiple of the time step is then written in the BRAM that stores the stimulation counters if the value is not zero.

Thus, all neurons can be stimulated independently with an independent period of time for a very low resource consumption and almost and no impact on the computation core performances. The timing diagram below shows the fram structure with stimulation duration noted as *stim dur*.



The stimulation duration for each neuron is stored in a BRAM that is read at each time step to detect if a stimulation is triggered in each neuron. Then, the stimulation duration is decremented by 1 and the updated value is stored in the BRAM. When, all the values were updated the module allows the update of the stimulation duration BRAM by the DMA from the PS. The Figure 3.10 illustrates the design of the external stimulation module in hardware.

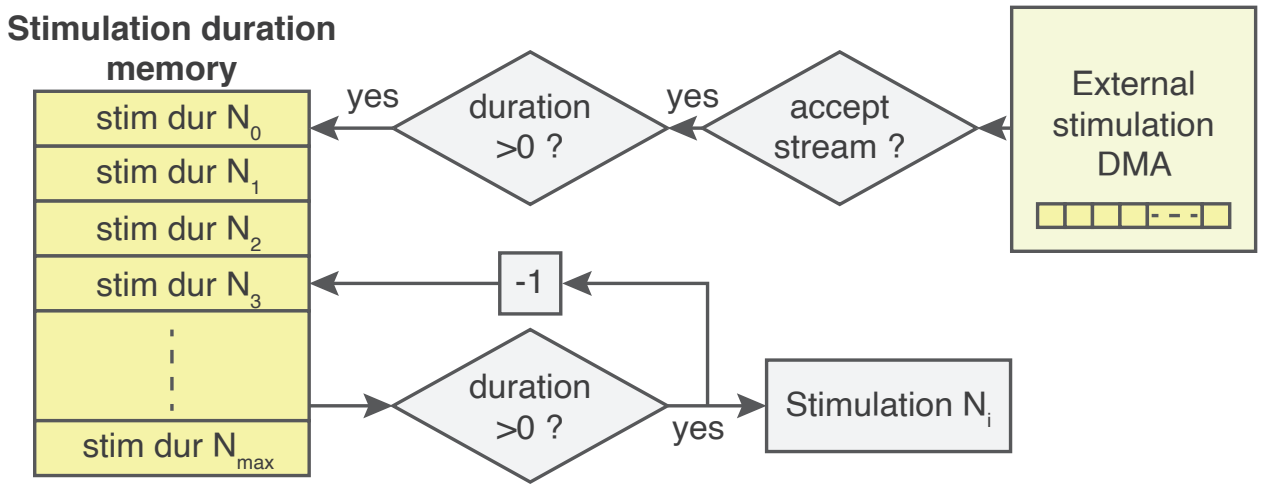


Figure 3.10: Block diagram of the external stimulation module in hardware. A stimulation trigger is applied to neurons for a stimulation duration given in number of time steps. The stimulation duration is set by stream received from DMA

3.5 Software: setup, control and monitoring by C++ application

The software setup and control on the SOM are carried out by a C++ application operating in the user space of the Canonical Ubuntu 22.04 for BicemuS. Using C++ allows a good compromise between performances and level of abstraction that can operate either with an operating system or in bare-metal using the Vitis tool chain, promoting versatility of the application to operate in different modes or on different targets. Additionally, the low-level of C++ suits well the proximity with hardware implied by FPGA.

This section presents the organization and main features of the C++ application that controls and set the network running in the PL part.

3.5.1 Structure and organization

The C++ application is operating in the user space of the Canonical Ubuntu 22.04 and uses the dma_proxy driver provided by AMD Xilinx to interact with the DMA s in PL. The application is responsible for the initialization, software and hardware configuration as well as monitoring. It can be executed either remotely *via* SSH or directly on the Ubuntu Desktop. The Figure 3.11 summarizes the different functions of the application.

Software configuration. The software configuration is performed from the software configuration file tuned by user in the JSON format that can also be generated by the Python scripts. It notably allows to enable the different monitoring channels as well as the main parameters of the application such as saving path or data collection intervals.

Hardware configuration. The hardware configuration uses the hardware configuration file generated by the Python scripts that contains all the parameters of the network. The hardware configuration set the parameters of the neurons, synapses, monitoring and fill the memory necessary for the network to operate.

Monitoring. The monitoring corresponds to 3 threads responsible for the waves and spikes monitoring as well as the external stimulation. The spikes and waves monitoring threads have similar architectures that start with a reading of the DMA data, then the frames are optionally saved locally in a file or sent via Ethernet. The external simulation wait for the stimulation trigger and data from Ethernet, then forward the stimulation information to the PL (see Figure 3.11).

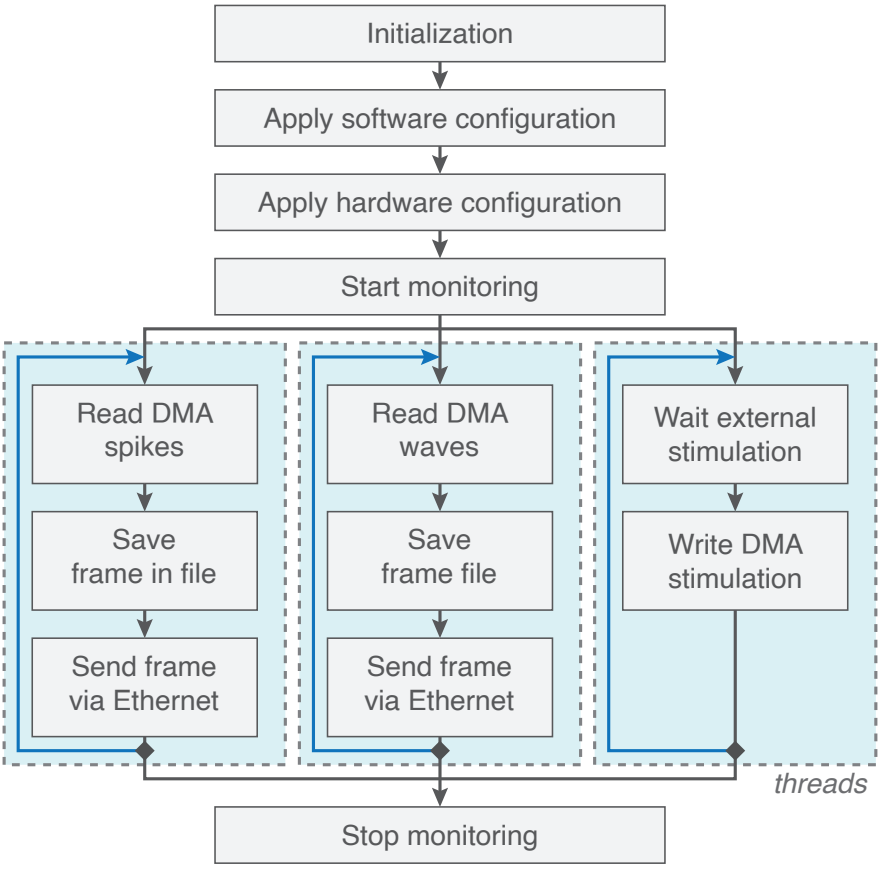


Figure 3.11: High-Level workflow of the C++ application showing the main functions. Monitoring of spikes and waves as well as external stimulation are operating in threads.

3.5.2 Software configuration

The software configuration is performed from the software configuration file in the JSON format that can be generated from the Python scripts or edited manually by the user. It participates in the flexibility of the system by allowing the user to adapt the system to the application by tuning data collection intervals or the monitoring channels to forward it to another system.

The parameters include the path to the hardware configuration file, emulation time, the configuration of the different monitoring channels as well as the stimulation step settings.

The selection of the different neurons to monitor on the DAC and DMA are set from a vector specifying their index. The DAC monitoring can monitor up to 8 waveforms while the DMA can monitor up to 16 waveforms. The saving path of both the waves and spikes monitored by DMA can be set to allow local storage on either the SD card or on any external device connected to board like a SSD drive connected *via* USB. The different monitoring channels can be enabled or disabled to offload the processors by allowing more CPU time to the other monitoring threads. A stimulation step occurring once can also be setup to specifying its duration and delay.

The parameters are parsed from the JSON file using a header only library "JSON for Modern C++" by nlohmann to simplify compatibility. The parameters allowing the configuration of the application are listed in Table 3.3.

Table 3.3: Configuration parameters of the C++ application setup from the JSON configuration file.

Key	Description
fpath_hwconfig	Path to hardware configuration file
emulation_time_s	Emulation time from 1 to 2^{32} s by 1 s
sel_nrn_vmem_dac	List of 8 neuron waveforms to select on DAC
sel_nrn_vmem_dma	List of 16 neuron waveforms to select on DMA
save_local_spikes	Enable/disable spike local saving
save_local_vmem	Enable/disable waveform local saving
save_path	Path to local saving file
en_zmq_spikes	Enable/disable ZeroMQ spike sending
en_zmq_vmem	Enable/disable ZeroMQ waveforms sending
en_zmq_stim	Enable/disable ZeroMQ external stimulation
en_WiFi_spikes	Enable/disable WiFi spike sending
ip_zmq_spikes	IP address ZeroMQ spike sending
ip_zmq_vmem	IP address ZeroMQ waveform sending
ip_zmq_stim	IP address ZeroMQ external stimulation
nb_tstamp_per_spk_transfer	Time stamps to wait for spike collection (x1 ms)
nb_tstep_per_vmem_transfer	Time steps to wait for waveform collection (x31.25 µs)
en_stim	Enable/Disable simulation step
stim_delay_ms	Stimulation step delay in ms
stim_duration_ms	Stimulation step duration in ms

3.5.3 Hardware configuration

The hardware configuration corresponds to the setup of the different properties of the network such as the HH neuron model, synaptic connections and weights, ion rate tables as well as the

monitoring properties. The setup is performed from the hardware configuration file generated by the Python scripts, then parsed by a custom parser.

The AXI-Lite communication is accessed using */dev/mem* that allows to control the registers as an array starting at the memory address of the AXI-Lite through a virtual pointer. The fixed-point and floating-point conversions are done using the custom functions.

Memories are initialized using the loopback register. The data, address and write enable are sequentially written. Then, the software waits for the loopback register to be equal to the address written to set the write enable low before moving on to the next address. This method is longer than a setup by AXI bursts but allows a simpler handling and resource utilization in hardware.

The hardware configuration also includes sourcing the seeds for the four noise generators in hardware to allow a good randomization of the noise in all runs.

3.5.4 Monitoring

The application handles the monitoring of spikes and waves using DMA and the external stimulation. The control of the DMA is performed by the dma_proxy driver provided by AMD Xilinx. Each monitoring channel runs in a thread to allow parallel monitoring with different options like on-board local saving as well as forwarding through Ethernet over ZeroMQ. The Figure 3.11 and Figure 3.12 summarizes the different monitoring options of the system.

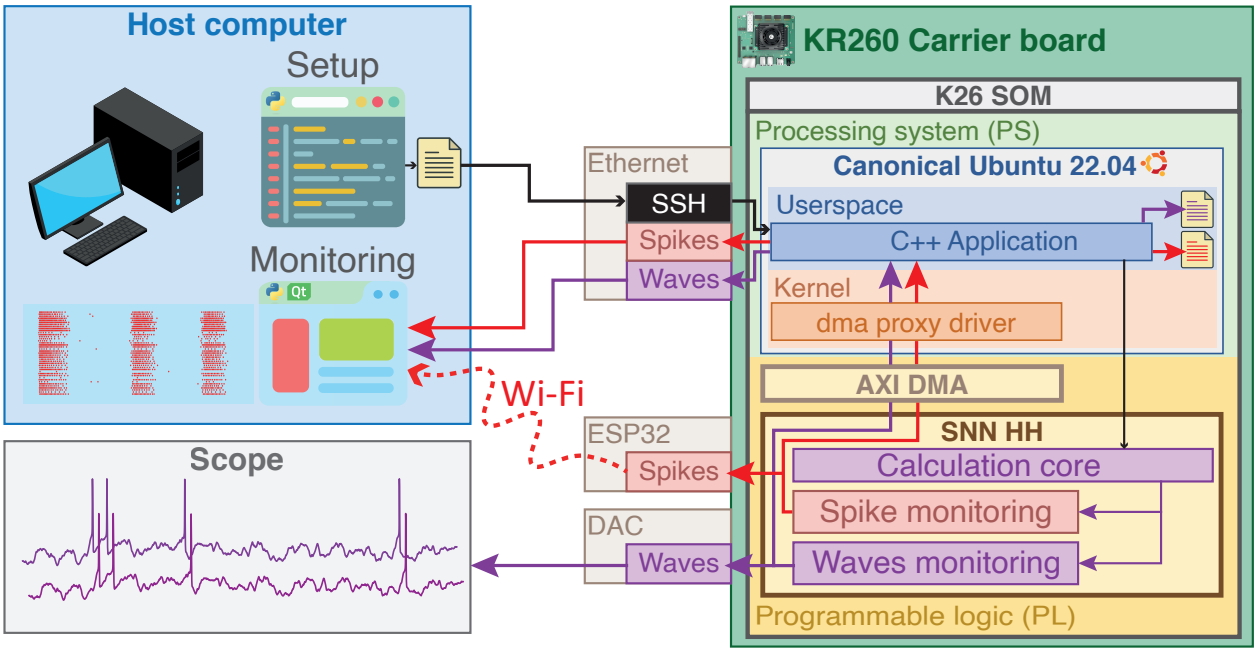


Figure 3.12: Overview of system setup highlighting the monitoring available and showing the setup from the configuration file generated by Python scripts.

ZeroMQ is an open-source universal messaging library widely used and embedded system to build efficient communication between applications. It offers different messaging patterns, networking functions and supports various programming languages and is often used for high-performance, low-latency communication in real-time applications. The choice of this library serves the purpose of versatility and performances of this application by providing performances, various programming languages as well as standardized communication.

Local file saving. The local file saving of spikes and waves corresponds directly to the data transferred by the DMA stored as binary without treatment to limit CPU involvement. This

method guaranties a fixed latency independent of the network activity that suits better real-time behaviors. While saving files with large buffers exhibits negligible latency compared to the time represented by the stored data, repeated writing of smaller buffers shows lower performance. Thus, larger data collection intervals show better performance for file saving and also reduce CPU involvement by limiting the frequency of file writing. Spikes can also be decoded and stored in csv format but shows fluctuating computation time depending on the network activity.

Ethernet forwarding. The Ethernet forwarding over ZeroMQ sends directly the data transferred by the DMA and can also operate with file saving. The communication pattern is PUSH/PULL that facilitates one-to-many distribution of messages so as the monitoring can be forwarded to many other systems.

External stimulation. The external stimulation also uses ZeroMQ to receive the stimulation frame in a PUSH/PULL pattern. It allows a standardized and easy interconnection with BiœmuS through the C++ application.

The performances of the monitoring can be improved by disabling local file saving or one of the threads to give more CPU time to the threads. Using the application remotely without the graphical interface of the desktop can allow more CPU time. A compromise can be found between the data collection interval and monitoring enabled so as multiple threads can operate concurrently efficiently if the data collection interval is large enough.

3.6 Software: configuration and monitoring by Python scripts

The Python scripts developed allow configuration and monitoring of the system. Python is a high-level open-source programming language known accessible and user-friendly that allows fast prototyping. Hence, it suits well the need for a flexible and user-friendly way to configure and monitor the system.

The Python scripts include the configuration scripts that generate the configuration files and emulate the configuration and the monitoring scripts that allow visualization of spikes or waves and send external stimulation.

The scripts can be executed either directly on-board or on another target, but the emulation may be slow and resource-intensive so better suited for powerful computers.

This section presents the functions and features of the Python scripts generating the hardware configuration file and software configuration file used to configure the system as well as monitoring.

3.6.1 Configuration

The configuration of the system through the Python scripts include all the parameters of the HH model, synaptic weights and types, ion table rates as well as various elements of the system.

The scripts implement preset types for neurons (FS, RS, IB, LTS) and synapses (AMPAR, NMDAR, $GABA_AR$, $GABA_BR$) using the parameters of [Pospischil et al., 2008] and [Destexhe et al., 1998].

The scripts are organized in classes that include *NeuronHH* for the HH neurons, *Synapses* for the different synapses and *HwConfigFile* that translates the configuration into the configuration file shown in Figure 3.13.

NeuronHH. The class *NeuronHH* set the parameters of the HH neuron model organized in types that are preset or can be created by the user. It includes the parameters of ion currents as well as generating the ion channel states tables from the functions and tune the size of the table. It also set other parameters such as the value of the stimulation current.

Synapses. The class *Synapses* set the parameters of the synapses model organized similarly to neurons. It includes the parameters of the synapses as well as other parameters such as pre-computed synaptic rates.

HwConfigFile. The class *HwConfigFile* converts the different parameters set by the user into the hardware configuration file that is used to set the system. This class is not intended to be modified by the user as it is linked with application.

The scripts can be executed either directly on-board or on another target. The Figure 3.13 shows the configuration scripts executed to generate the configuration files either on a computer or directly on-board.

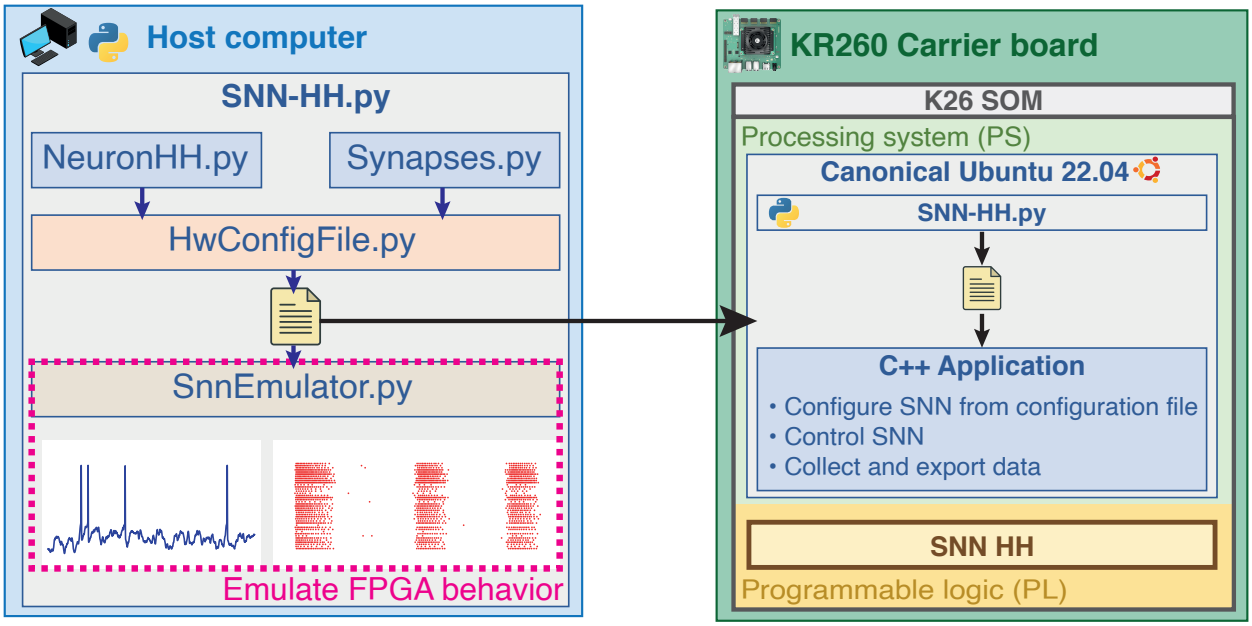


Figure 3.13: Overview of system setup from the configuration file generated by Python scripts ran either on-board or on another computer. The configuration file is then read by a C++ application running on Canonical Ubuntu operating system in the PS part to set up the SNN in PL part.

3.6.2 Emulation

Along with the configuration scripts, comes a class allowing the emulation of the configuration file generated to assess the behavior of the network beforehand as shown in Figure 3.13. It allows emulation of the network using the Forward Euler, solver, ion channel states pre-computed tables as well as fixed-point coding.

SnnEmulator. The class *SnnEmulator* allows the emulation of the configuration file generated by running a simulation using the parameters from the file. It allows visualization of the internal variables of the simulation like ion channel states tables, currents or membrane

potentials. It features different plotting types like raster or membrane potentials subplots to verify the behavior configured.

The emulation is based on commonly used libraries like the *numpy* library for calculation, *Fxpmath* library for fixed-point coding and *matplotlib* for graphics to provide good performances while keeping the code accessible. Nonetheless, the emulation is not optimized so as the computation time is quite high for large network or long emulation times.

3.6.3 Monitoring

The monitoring scripts correspond to graphic user interfaces that allow to monitor spikes or waves using Qt library with Python. PyQt is a set of Python bindings for the Qt application framework enabling cross-platform Graphical User Interface (GUI) and applications using Python.

It provides easy integration of Python with the rich features of Qt, allowing interactive interfaces that can handle events. GUI development is notably fast to develop and show satisfying speed and responsiveness as Qt has an optimized underlying C++ architecture. An example of spike monitoring interface is shown in Figure 3.14.

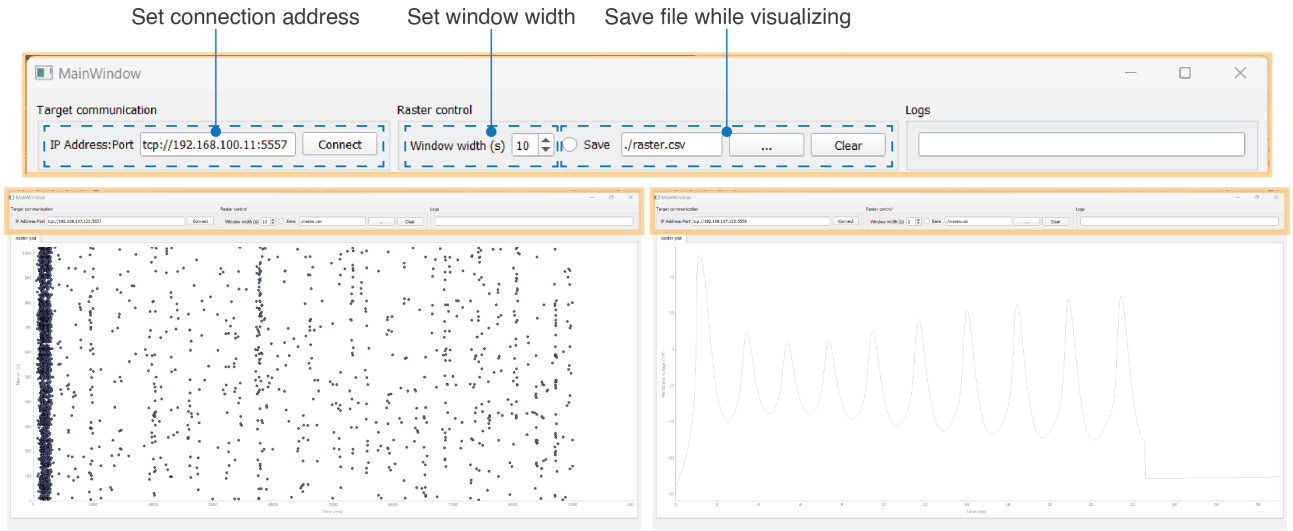


Figure 3.14: Qt-based Graphical User Interface (GUI) developed for spikes or waves monitoring.

The monitoring interfaces are based on *Qthread* that creates and manages threads in a Qt application to perform tasks concurrently without blocking the main user interface thread. The different threads communicate with each other using *pyqtsignal* that can notify a thread and carry the data to the function connected to the signal.

These classes allow facilitated handling of usually complex processes in C/C++ while maintaining correct performances, making it more accessible for users to add features to the interface. Examples of features added to the monitoring interfaces will be presented in the next chapter that focuses on applications.

3.7 Performances

The system designed is operating on a small and cost-optimized target so that it is important to assess the efficiency of the solution by analyzing the performances of the system in various aspects such as the resource utilization, power consumption and latency. These metrics will

allow comparison with the existing system in literature, localize bottlenecks, find optimizations and consider using a smaller target for an alternative version.

This section presents the performances of BiœmuS in terms of resource utilization, power consumption and latency.

3.7.1 Resource utilization

The resource utilization presented correspond to the implementation of BioemuS on the AMD Xilinx KR260 Robotic Starter Kit featuring 1,024 neurons with 6 conductance-based currents and a total of up to 2^{20} conductance-based synapses for a time step of 31.25 µs. The resource utilization report associated with this implementation is shown in Figure 3.15.

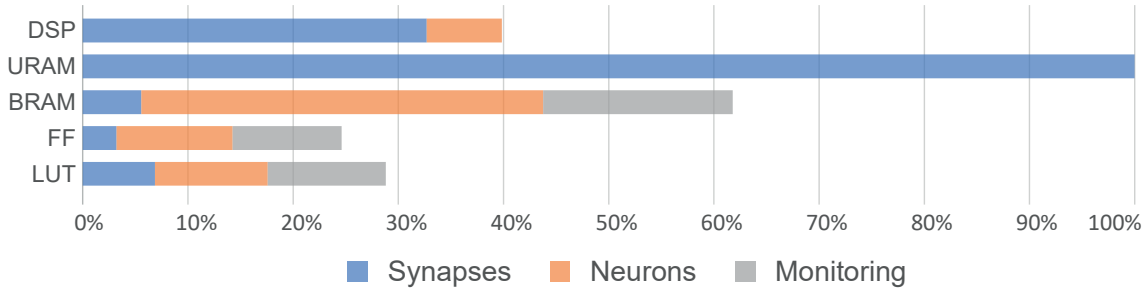


Figure 3.15: Distributed resource utilization of BiœmuS for implementation on AMD Xilinx KR260 Robotic Starter Kit exported from Vivado 2022.2.

The part of the system that consumes the most resources corresponds to the synapse block. The synapse block is the most challenging part of the system as it represents a significant part of the computation and storage. To obtain the best performances in this critical part, it is required to provide sufficient throughput that imply high parallelization of the computation and storage elements, thus highly impacting the resource utilization. As this implementation is fully connected, the resource consumption is then at a maximum.

The resource utilization footprint of neurons is corresponds mostly to the BRAM used for the parameters storage as well ion channel states memories. Using smaller ion channel states table could allow the implementation in LUTRAM, thus highly reducing the number of BRAM used. The FF and LUT usage corresponds mostly to the logic implemented parts of floating-point calculation like additions and subtractions, pipeline registers and control. The higher resource utilization of the floating-point coding is compensated by the greater flexibility brought to the system.

The resource utilization can vary depending on the size of the monitoring elements so as modifying the size of the monitoring FIFO or number of neurons to monitor can increase the resource utilization. Especially, the FIFO in packet mode for spikes and waves monitoring as well as DMA are using a quantity of BRAM depending on the size of the tranfers.

While most of the memory available is used, less than 50% of the computing capacity (Logic and DSP) of the board is used by the system (see Figure 3.15). As the design is implemented on an entry level target, the projection of the resource utilization on larger targets suggests the possibility to run several calculation cores in parallel (see Figure 3.16) as well as allowing faster emulation.

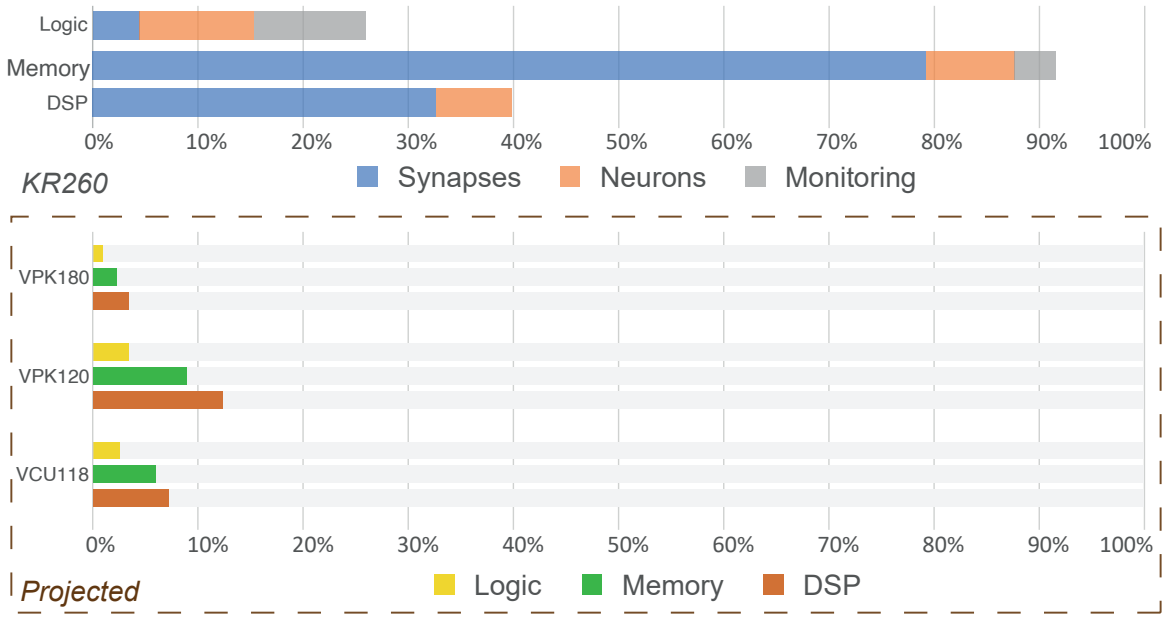


Figure 3.16: Resources utilization of BiœmuS. Utilization for main modules implemented on AMD Xilinx KR260 Robotic Starter Kit and projected on high-end evaluation boards from AMD Xilinx (Versal Premium Series VPK120 and VPK180 Evaluation Kits and Virtex UltraScale+ VCU118 Evaluation Kit). Logic corresponds to LUT and Flip-Flops, memory to the total memory implemented as BRAM and URAM, DSP to the number of DSP slices.

An important element to consider is that the recent architecture such as Versal Adaptive SoC by AMD Xilinx shows significantly better support for floating point calculation such as native support in DSP for most operations. Hence, the implementation of the system on such target is most likely to reduce considerably the resource utilization and optimize the performances.

3.7.2 Power

The power consumption is an important metric of the system for embedded systems, especially in our applications that are in the end aiming to be embedded in neuroprostheses or other electroceutic devices.

The overall system power consumption is 6.50W with 3.42W associated with the calculation core as shown in Figure 3.17. Considering only the calculation core that is running on PL part, BiœmuS consumes 3.42 times more than SpiNNaker [Painkras et al., 2013] or BrainScaleS-2 [Pehle et al., 2022] that run on ASIC.

While it is known that Application-Specific Integrated Circuit (ASIC) usually consumes less power that FPGA, it is more likely that our system consumes more power as the design prioritized flexibility over power efficiency. The power consumption could be optimized by updating the system with power efficiency in mind. Adding control design disabling the BRAM or URAM when not used or accurate estimation of the monitoring size needed could reduce the power consumption, but it could also translate in more constraints on the routing that could affect the timing and resource utilization.

The development of a truly power-efficient system would rather rely on the choice of a smaller target and using a reduced version of the system prioritizing power consumption over flexibility and performances.

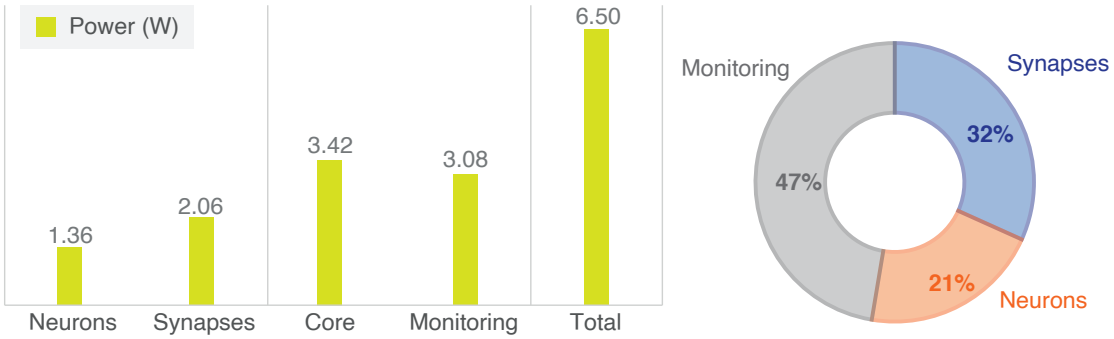


Figure 3.17: Distributed power consumption for implementation on AMD Xilinx KR260 Robotic Starter Kit exported from Vivado 2022.2.

3.7.3 Latency

Latency is crucial information in real-time system. Applied to the computation core, the latency for the calculation of all the membrane potentials, the latency has to be inferior to the time step of 31.25 µs to maintain real-time behavior.

As for monitoring, it has to be kept as low as possible to obtain coherent interfacing with biology. BiœmuS allows different monitoring solutions in order to cover many interfaces and compromises between throughput and latency to better suit the applications.

Local saving DMA. The on-board saving in file allows obtaining relatively low latency as the latency roughly corresponds to the DMA read and write operations through the dma_proxy driver.

The latency observed to save spikes in file using binary format is displayed in Figures 3.18A1,A2. As for instance from Figures 3.18A2, in most cases the latency to save 100 ms of spikes is between 69 and 125 µs. The main drawback of this monitoring is that it enforces offline analysis.

Ethernet ZeroMQ. The monitoring using Ethernet over ZeroMQ shows a good compromise between throughput and latency. The latency of the Ethernet communication is low so as the average latency observed to send spikes through ZeroMQ (UDP) is evaluated to 240 µs for 100 ms of spiking activity based on ZeroMQ latency tests, thus showing a good ratio between the latency and the amount of data transferred.

The complete latency of this monitoring channel includes the fluctuating latency read and write operations of the DMA using the dma_proxy driver.

The latency observed for the application to send the data over ZeroMQ is displayed in Figures 3.18B1,B2. From the chart presetend in Figure 3.18B2, the latency evaluated to send 10 ms of spikes is between 10 and 25.75 µs in most cases and between 18 and 43 µs for 100 ms of spikes.

Analog waves using DAC. The monitoring of analog waves using the DAC is the monitoring channel with the smallest latency as it purely hardware. The latency to update the value of all channels of the DAC is about 6 µs.

Nonetheless, the accuracy of this monitoring is limited as it is only 12-bits DAC and the throughput is quite low as it can only output 8 neurons.

Wi-Fi using ESP32. The monitoring Wi-Fi includes the latency of the SPI transfer to the microcontroller that is very low as the DMA of the microcontroller with a very light software layer and the latency of the Wi-Fi that is quite high. Indeed, the latency of the Wi-Fi protocol is known to be quite high and in the millisecond range.

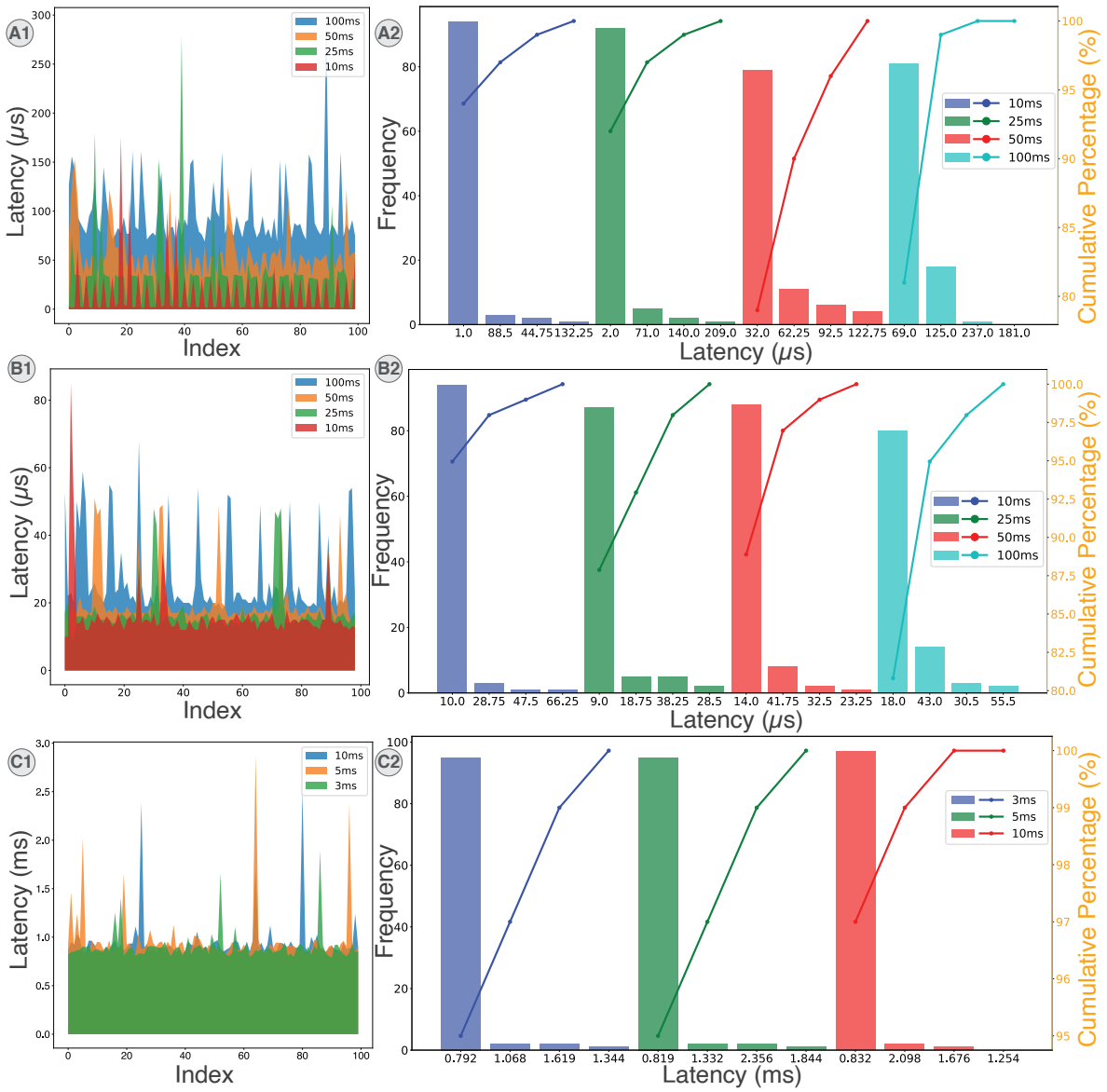


Figure 3.18: Latency charts for spike monitoring for 100 data collections at different intervals. **(A1,A2)** On-board file saving latency for spikes coded in binary based on 100 samples at different data collection intervals represented sequentially and in Pareto charts. **(B1,B2)** Latency in application to send spikes with ZeroMQ over Ethernet for spikes coded in binary based on 100 samples at different data collection intervals represented sequentially and in Pareto charts. **(C1,C2)** Latency to send spikes over Wi-Fi between two ESP32 for spikes coded in binary based on 100 samples at different data collection intervals represented sequentially and in Pareto charts.

The latency between two ESP32 was evaluated over 100 data collection and different data collection interval and is summarized in Figures 3.18C1,C2. From Figure 3.18C2, the average latency observed for spike monitoring through Wi-Fi (UDP) between two ESP32 is about 0.792 ms to 2.098 ms depending on the data collection interval ranging from 3 ms to 10 ms.

The main benefit of this monitoring is the wireless nature that suits well embedded system, but that comes with the drawback of higher latency compared to the other solutions.

To put if briefly, the monitoring latency is an essential element of the system as it monitors the computation core that operates real-time. The monitoring developed show compromises bases on the monitoring channel used.

3.8 Summary

This section detailed the complete design of the system developed while supporting the choices made to build a flexible real-time biomimetic SNN with accessible control and monitoring. It detailed the different hardware components responsible for computation of the artificial neural network and its monitoring while explaining the main limits and strength of the design. It also introduced the software developed to control, setup and monitor the system through a C++ application that provide performances and flexibility. Finally, it introduced the Python scripts developed to generate the configuration files as well as monitoring the system using Python in a user-friendly way. In conclusion, this chapter introduced the system designed as well as its strength and limits that will be exploited in the next section treating about the applications of BiœmuS.

4 Applications and biohybrid experiments

4.1 Introduction

Having meticulously introducing the basic knowledge in biology, presented the target used and detailed the architecture of the system, the applications enabled by the system as well as the results obtained can be demonstrated.

The BiœmuS system enhances its potential from practical applications such as real-time emulation and biohybrid experiments as a tool for neuroscientists to study neurological disorders. The applications also showcase how the system integration promotes versatility and ease of use so as it is accessible among biologists. The prototype BiœmuM also demonstrates its capacity to perform real-time emulation of multicompartment neurons.

This chapter demonstrates applications that use BiœmuS as a real-time emulator of biomimetic networks to create a fast emulation setup for large biophysically detailed network. Then, it presents the biohybrid experiments conducted using the system that shows how different network implementation from single neuron to larger network can interact with biology through various interfaces. Additionally, the alternative versions of the system showing different compromises in terms of energy and embedding considerations. The Figure 4.1 illustrates the possible applications of the system.

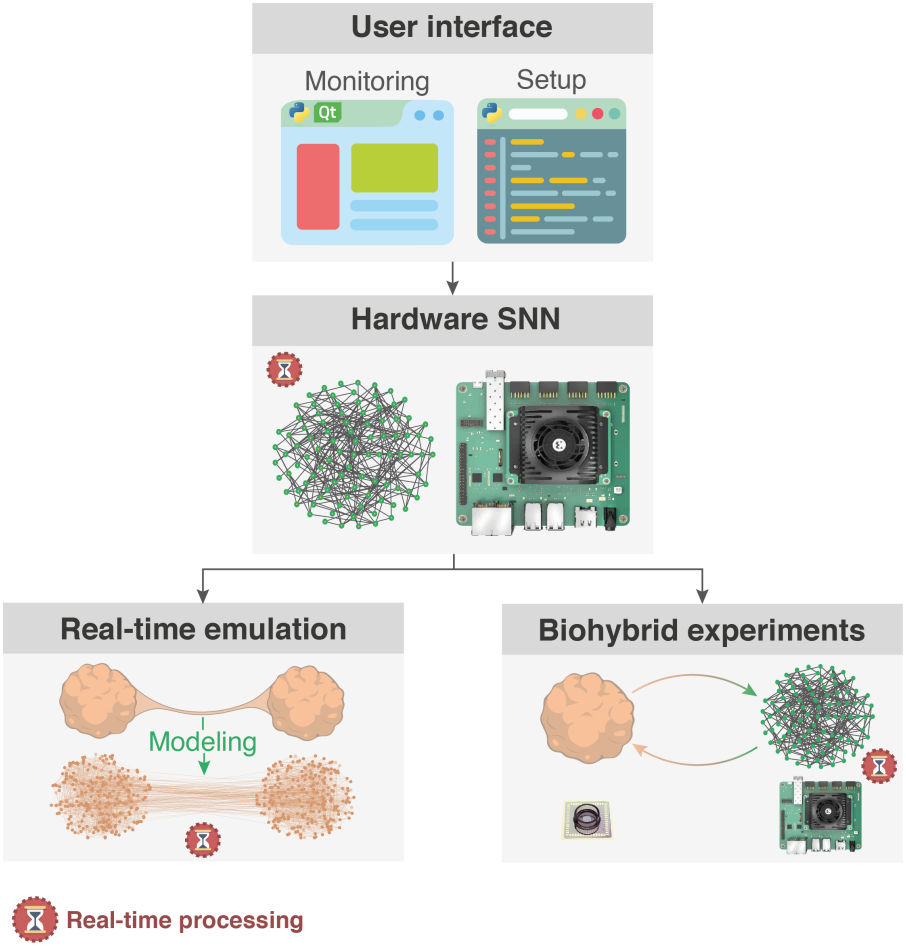


Figure 4.1: Overview of system applications. The real-time biomimetic SNN (BiœmuS) implemented in hardware is monitored through the Qt-based GUI and setup by Python scripts ran either on-board or on another computer. The SNN is used either as a real-time emulator for biophysically realistic models or integrated in a biohybrid experiment setup. In a real-time emulation setup, it runs fast simulations of biophysically detailed models suited for large parameters sweeps. Integrated in a biohybrid experimental setup, it acts as a versatile biomimetic artificial neural network easily interfaced with standard biological recording units.

4.2 Real-time emulation of biophysically detailed neurons

To begin with, real-time emulation of independent neurons emphasizing biophysical detail will be presented. This section presents real-time emulation of biomimetic single compartment neurons capable of generating spontaneous activity using BiœmuS as well as multicompartmental modeling of motor neuron using BiœmuM.

4.2.1 Spontaneously spiking single compartment neurons

The spontaneous activity of neurons is an essential property of neural networks. This ability to generate action potentials without external stimuli is modeled in the system by the introduction of a current that mimics synaptic noise [Destexhe et al., 2001, Khoyratee et al., 2019]. By modifying the parameters of the equation ruling the synaptic noise, different level of spontaneous activity can be reproduced. As for instance, it allows to match the dynamics of biological culture of neurons to obtain a coherent model.

In the main version of BiœmuS that emulates single compartment modeling in real-time, the activity of neurons can be monitored online through waveforms or spiking activity. The Figure 4.2 shows an example of a configuration implementing 1,024 spontaneously spiking FS neurons.

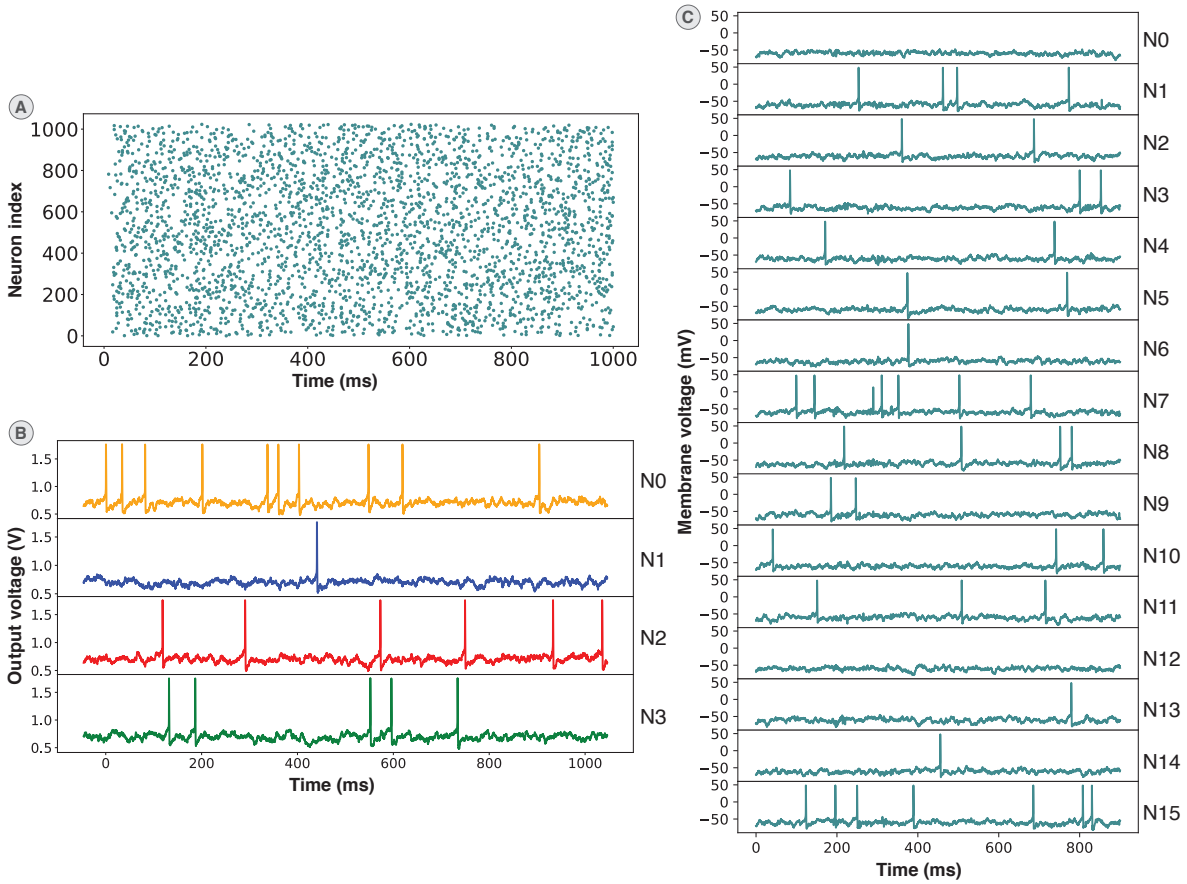


Figure 4.2: Emulation of independents FS neurons with synaptic noise monitored with the on-board saving and DAC output. **(A)** Spiking activity of 1,024 FS neurons monitored using the local-saving of data transferred with DMA. **(B)** Waveforms of the membrane voltage of 4 FS neurons captured using a 4-channel oscilloscope from the output of the DAC PMOD. **(C)** Waveforms of the membrane voltage of 16 FS neurons monitored using the local-saving of data transferred with DMA.

The most efficient way to monitor the activity of the spontaneously spiking neuron is shown in Figure 4.2A that displays the raster plot of the 1,024 neurons emulated. This representation allows validating the spontaneously spiking ability of the neurons emulated based on the observation of independently spiking neurons that does not show synchronized activity. Hence, the synaptic noise injected successfully created different spontaneous activity for each neuron emulated.

Along with the spiking activity, the membrane potential of neurons showing the noisy dynamics of the neurons are shown in Figures 4.2B,C.

In Figure 4.2C, the membrane potential of the 16 neurons monitored using the DMA is presented, validating the independent noise generation as the independents neurons show different spiking activity while being modeled with the same parameters. This monitoring channel is the most reliable for the Waveforms as the membrane potential monitored is captured at each step on 32 bits.

Another possible monitoring of the membrane voltage that is less accurate is the DAC output as shown in Figure 4.2B. The DAC allows visualization of the membrane potential of neurons at the maximum sampling frequency of system (2^{-5} ms, i.e. 32 kHz), but only in a 12-bit coding translated in a voltage between 0 and 2.5 V. Up to 8 neurons can be monitored per DAC PMOD, but only 4 are displayed in Figure 4.2C as the recording oscilloscope only featured 4 channels.

4.2.2 Multicompartmental motor neurons to study ALS

Amyotrophic Lateral Sclerosis (ALS) is a rapidly progressive neurodegenerative disease that targets motor neurons. It is one of the most common and devastating neurodegenerative disease.

In order to study the disease, one of our collaborating team developed models of motor neurons affected by ALS, specifically motor neurons of SOD1 mice at embryonic state [Branchereau et al., 2016, Branchereau et al., 2019, Martin et al., 2020]. The models were developed with a high level of biological meaningfulness. The dynamics of the neurons were reproduced accurately thanks to a modeling based on patch clamp recording of each ion channels. Most notably, the model include a highly detailed modeling of the morphology of the neuron, thus enforcing the use of a multicompartmental modeling.

The models were developed using the NEURON software so as a real-time emulation of a network is less likely to be possible, hence prompting the development of BiœmuM. BiœmuM is a prototype version based on BiœmuS developed in this work aiming to emulate multicompartmental neurons.

The model used for this application is the motor neuron at day E13 presented in [Branchereau et al., 2016] that implements 133 segments (or compartments) distributed in soma, axons and dendrites sections based on patch-clamp recordings. The morphology of the neuron generated from the NEURON model is presented in Figure 4.3A. The currents involved in the model are the potassium, sodium and leakage currents that show different conductances depending on the section. As for instance, only the active axon and the rest of the axon integrate sodium current. The Figure 4.3B recapitulates the morphology of the neuron and shows how the sections are connected.

As the first iteration of BiœmuM is capable of emulating up to 64 segments, the model was reduced to a total of 64 segments while preserving the sections and their interconnections. The simplified model was compared to the original model in the NEURON software in response to a stimulation of 15 ms inserted in the soma to assess the coherence of the simplified model as

shown in Figure 4.3C. While the simplified model is not capable of accurately reproducing the spatial morphology of the neuron, its accuracy remains satisfying in this application aiming to validate the prototype system BiœmuM. Indeed, the membrane potential of the simplified model is closed to the original at two distant points being the soma, where the stimulation is inserted and at the end of the axon.

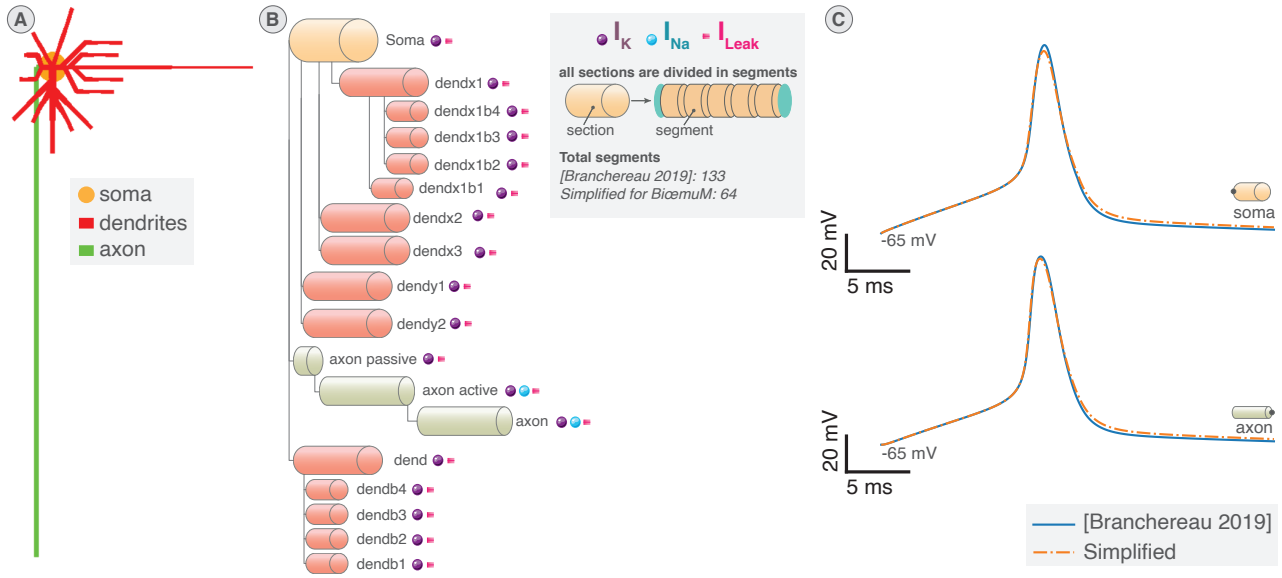


Figure 4.3: Simplification of the multicompartmental model of motor neuron at day E13 from [Branchereau et al., 2019] using NEURON software. **(A)** Morphology schematic of the motor neuron at day E13 showing. **(B)** Morphology of the neuron decomposed in sections of varying geometrical and electrical properties (length, diameter, ion and leakage currents). Sections are decomposed in fewer segments (or compartments) in the simplified modeling. **(C)** Comparison of the evolution of the membrane potential in response to a 15 ms stimulation pulse inserted in the soma. Membrane potentials are recorded in the soma and at the end of the axon.

The simplified model was then translated to the BiœmuM python scripts that allows to emulate in software the behavior of the configuration file generated. The configuration file generated was then ran using BiœmuM and the membrane potentials of the 64 segments were monitored using the local file saving through DMA.

The membrane potentials obtained were then compared to the software emulation as shown in Figures 4.4A,B. The Figure 4.4A presents the 64 segments of a neuron overlapped, showing that the action potential is almost identical in all compartments except for the first segments of the axon and that software and hardware emulation match. The Figure 4.4B shows the membrane potentials arranged by segment index for both software emulation and hardware emulation, allowing visualization of all membrane voltages and the fitting of the hardware emulation with the reference. Hence, these results validate the implementation of the prototype system BiœmuM and demonstrate its ability to emulate multicompartmental neurons in real-time.

The resource utilization of BiœmuM for an implementation on the AMD Xilinx KR260 Robotic Starter Kit is shown in Figure 4.5. The main differences in the resource utilization of BiœmuM compared to BiœmuS are explained by the absence of synapses, full floating-point coding, different solver and waveform oriented monitoring.

Since BiœmuM focuses more on the membrane potential of the neuron, it enforces the need of larger monitoring of the membrane potential, translated here by a larger amount of BRAM used for the monitoring of the waveforms. Additionally, the multicompartmental model involves

more parameters to store for each neuron that is translated in more BRAM used for the HH coefficients storage. Also, as BiœmuM is operating completely in single floating-point coding, the implementation cost for the HH coefficients is significantly higher in DSP, FF and LUT. Moreover, the matrix solver, enforced by the use of the multicompartmental model, requires a large amount of BRAM used as buffer and context memory, increasing with the number of neurons (Matrix Solver part in Figure 4.5).

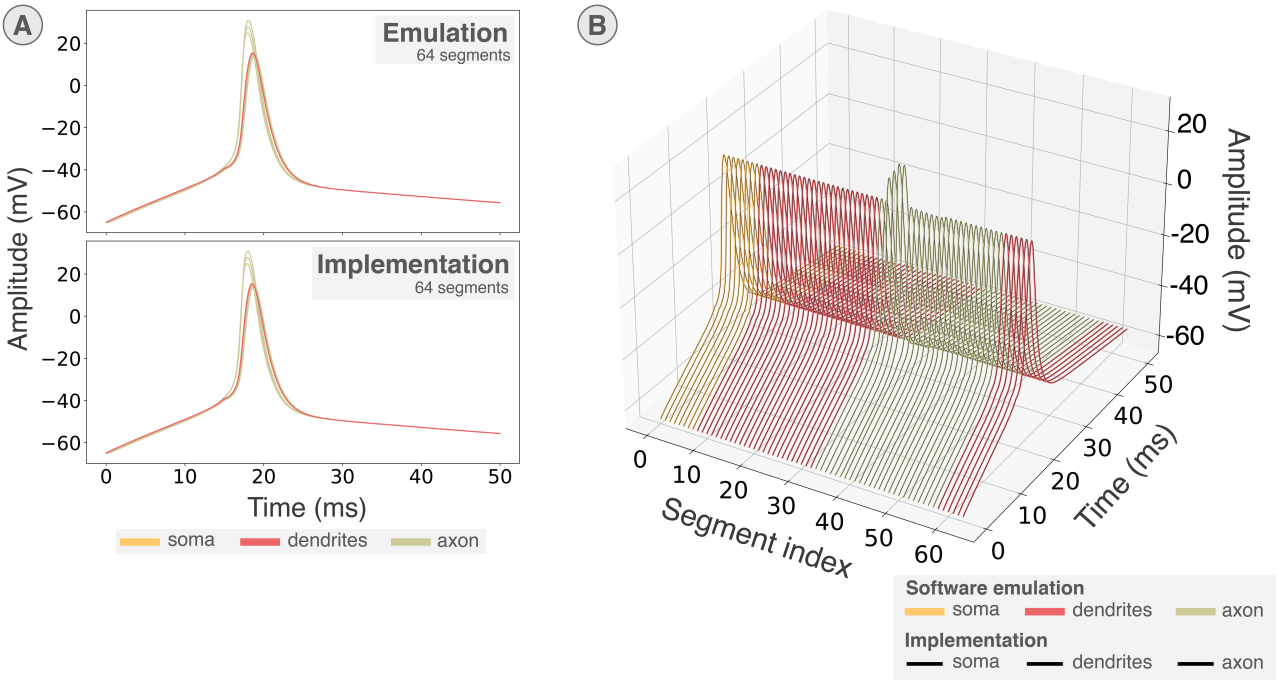


Figure 4.4: Comparison of membrane potentials in software emulation through the Python scripts and hardware implementation. Membrane potentials in implementation were recorded using the on-board file saving through DMA. **(A)** All 64 segments overlapped in both emulation using the Python scripts of BiœmuM and implementation on KR260. **(B)** All 64 segments sorted by segment index for both emulation using the Python scripts of BiœmuM and implementation on KR260.

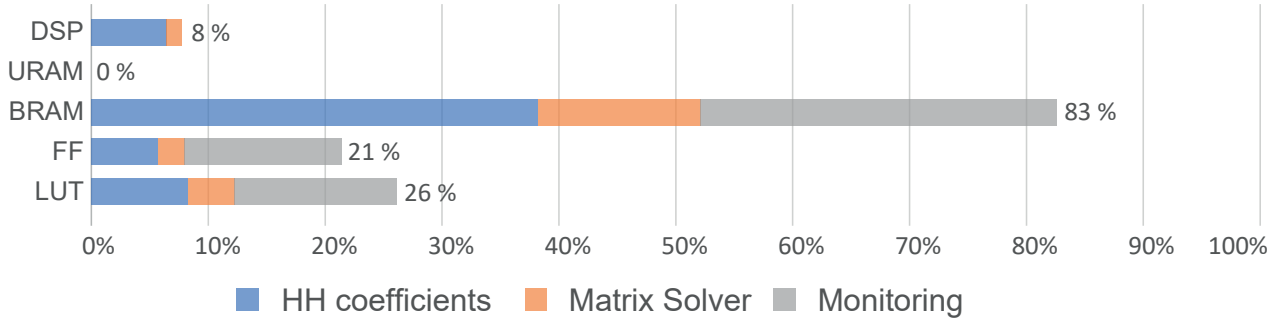


Figure 4.5: Distributed resource utilization of BiœmuM for implementation on AMD Xilinx KR260 Robotic Starter Kit exported from Vivado 2023.1 for 16 neurons of 64 segments. *HH coefficients* corresponds to the computation of the ion currents of the HH model and the storage of its parameters, *Matrix solver* to the solver computation paired with the context and buffer memory and *Monitoring* to the DMAs and buffer memories related.

This first iteration of the BiœmuM system shows promising results supported by the capability of emulating up to 16 neurons of 64 segments each parallelly in real-time where current

solutions are showing much higher computation times. As for example, the software emulation of 1 second of the simplified model in NEURON takes an average 3.5 seconds for 1 neuron on an Intel i7-10875H. Nonetheless, as multicompartmental modeling is getting a growing interest in recent years, more and more GPU implementations of significantly larger size and improved computation time are released [Stimberg et al., 2019, Kobayashi et al., 2021, Zhang et al., 2023]. The Table 4.1 presents a brief comparison of the performances of the system to two implementation that represents well the order of magnitudes found in GPU and CPU implementations.

Table 4.1: Comparison of some recent multicompartmental implementations of HH neurons on GPU and CPU.

	BiœmuM	[Kobayashi et al., 2021]	[Mäki-M et al., 2018]
Architecture	FPGA	GPU	CPU
Model	HH	HH	HH
Neurons	16	3072	150
Segments	64	674	80
Synapses	/	780,404	∼2500
Computation time (ratio)	1	9000 (1 s for 2.5 h)	828 (10 s for 2.3 h)
Target	SOM K26	Tesla V100	1 core

The Table 4.1 clearly highlights the main drawbacks of the current prototype of the system that are the absence of synapses and the relatively low number of segments. However, as BiœmuM shares the same base as BiœmuS, fully connected conductance-based synapses could be implemented while preserving the real-time operation of the computation core. As for the low number of neurons and segments, it could be largely increased by porting the system on a larger target with a more recent architecture like AMD Versal Adaptive SoCs that provide better floating-point support and larger resources available. Additionally, Versal architecture would allow to greatly improve the number of segments through faster clocking and optimized floating-operations, the main bottleneck of the system being the computation latency induced by the solver iterations equated in Equation 4.1.

$$max_{nb\ segments} = \frac{dt \times f_{clk}}{lat_{load\ context} + lat_{backward\ sweep} + lat_{forward\ sweep}} \tag{4.1}$$

where, $max_{nb\ segments}$ is the maximum number of segments that can be implemented for one neuron, dt the time step, f_{clk} the clock frequency, $lat_{load\ context}$ the latency to load the solver context, $lat_{backward\ sweep}$ and $lat_{forward\ sweep}$ the latencies to compute one operation of the backward and forward sweep.

Nonetheless, the strength of the system relies on its real-time computation and versatile interaction as it shares the same integration as BiœmuS. As for example, the waveform monitoring using the DAC would allow monitoring up to 8 membrane potentials in real-time. Furthermore, considering the affordable price of the target along with the performances obtained for an entry-level FPGA, the prototype version shows promising preliminary results. Finally, the most important point of the system is the real-time emulation that is a crucial requirement for the realization of electroceutic therapies, making of this system a novel tool to drive stimulation at a higher level of biological meaningfulness though the use of multicompartmental model.

4.3 Real-time emulation of complex neuronal structures

This section presents an application of the system to perform real-time emulation of complex network model representing three-dimensional tissue cultures that are derived from stem cells, human cerebral organoids, that are biological models of human brain.

This application demonstrates an example of real-time emulation of a model to provide a tool enabling fast emulation to predict and investigate the behavior of biology.

As such cultures are known costly and time-consuming, having such a tool could help neuroscientists to better orientate their experiments based on simulations.

4.3.1 Interconnection of human cerebral organoids

As a biological model to study human brain and the interaction between the different regions of the brain, three structures of interconnected human cerebral organoids were designed.

Human cerebral organoids are widely used to study human brain by reproducing structures of certain brain area [Kim et al., 2020]. The cortical organoids were generating using the reported protocol [Osaki and Ikeuchi, 2021]. Briefly, hiPSCs are dissociated then seeded into well plates until complete formation of the organoids thanks to induction of various media based on the culture time.

The three structures introduced by the model are *single*, *assembloid* (or fused) and *connectoid* that each promote different synaptic connections between the organoids so as the *connectoid* is aimed to show stronger activity than the *assembloid*. The different structures are illustrated in Figure 4.6.

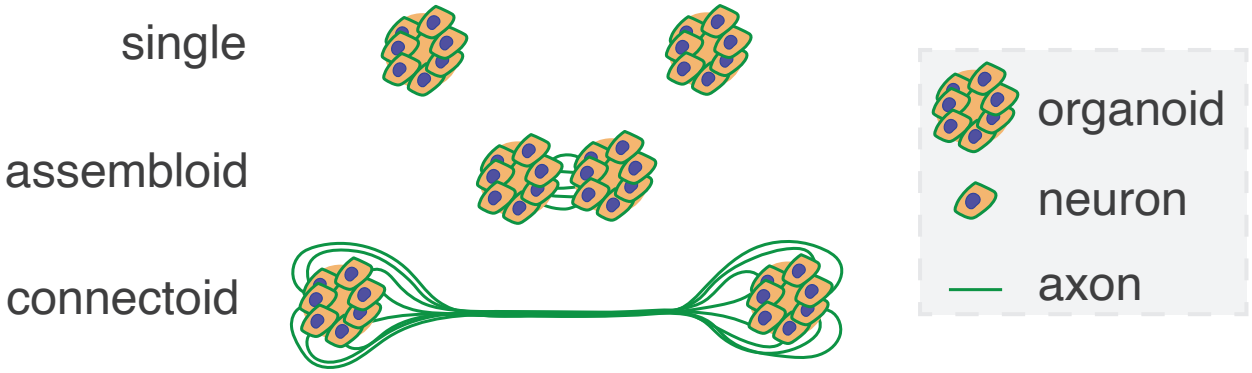


Figure 4.6: Illustration of the three interconnection structures of human cerebral organoids interconnections.

Single. The *single* physically separates the organoids to prevent connection between organoids. It acts as a reference model showing activity of independents organoids.

Assembloid. The *assembloid* structure, or fused, places organoids close to each other thus favouring connection of neurons based on proximity [Paşca, 2019], so as connections are mostly formed at the interface between organoids.

Connectoid. The *connectoid* structure places organoids centimeters apart while constraining the interconnection to form an axon bundle connecting mostly neurons on the surface of the organoids [Kirihara et al., 2019, Kawada et al., 2017], thus creating the strongest connection amongst the three structures.

4.3.2 Artificial modeling

In order to artificially model the three structures, it is first necessary to accurately model a cerebral organoid. Suitable types of neurons include FS and RS neurons that actually model cortical neurons. As for synapses, excitation and inhibition are both necessary to replicate the network activity. AMPAR and $GABA_AR$ are synaptic receptors found in cerebral organoids respectively responsible for fast excitation and fast inhibition.

The second step of the modeling concerns the reproduction of the different interconnection properties based on the spatial distribution of synaptic connections. In order to introduce a spatial dimension for the neurons in the network, neurons were assigned XY coordinates normally distributed. Then, the synaptic connections between neurons were generated based on probabilities following different functions based on the biological observations.

The synaptic connection rules for the synaptic connections inside organoids are ruled by Equation 4.2 that favors connection to neurons close to each other normalized by the diameter of the organoid. The connections between organoids are ruled by Equation 4.3 for *assembloid* and by Equation 4.4 for *connectoid*. The former favors connection to neurons close to each other normalised by the maximum distance possible between neurons, while the *connectoid* rule is promoting connection based on the location of neuron in the organoid that promotes connection on the exterior ring.

$$p_{single} = p_{max} \times (1 - \frac{d_{n_{pre},n_{post}}}{r_{org}}) \tag{4.2}$$

$$p_{assembloid} = p_{max} \times (1 - \frac{d_{n_{pre},n_{post}}}{d_{org_{pre},org_{post}} + r_{org_{pre}} + r_{org_{post}}}) \tag{4.3}$$

$$p_{connectoid} = p_{max} \times \frac{1}{2} \times (\frac{d_{n_{pre},org_{pre}}}{r_{org_{pre}}} + \frac{d_{n_{post},org_{post}}}{r_{org_{post}}}) \tag{4.4}$$

where p_{max} is the maximum probability of connection, d is the distance, $diam_{org}$ the diameter of the organoid, r the radius, n_{pre} and n_{post} the pre-synaptic and post-synaptic neurons, org_{pre} and org_{post} the pre-synaptic and post-synaptic organoids and the distance calculated from the center of the organoids.

The Figure 4.7 presents heatmaps of the number of synaptic connections per number associated with their XY coordinates based on the average of 40 random distributions.

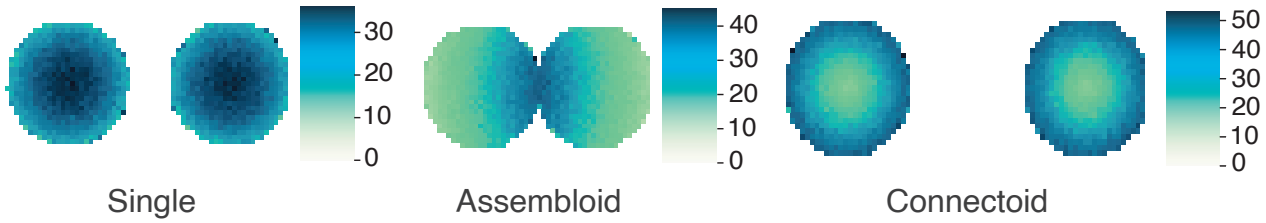


Figure 4.7: Heatmap of the number of connection per neuron for the three structure of organoid interconnections. *Single* corresponds to the synaptic connection inside both organoids. *Assembloid* corresponds to a higher synaptic connection at the interface between the organoids. *Connectoid* favors synaptic connection on the exterior ring of the organoid. Heatmap are based on an average of 40 random generation of synapses for a given XY mapping.

The heatmaps presented in Figure 4.7 shows higher connection at the interface between the organoids for the *assembloid* and on the surface of the organoids for the *connectoid*. The

synaptic connections inside the organoids, represented by the *single* structure, shows an almost uniform distribution except for the edge explained by fewer surrounding neurons. Hence, these results validates the rules synaptic connection rules.

The generation of the configuration file for this model was performed by adding a new Python class that simply assign normally distributed XY coordinates to neurons and generate synaptic connections based on specific rules for each structure. The class implemented handles neurons and synapses simply as excitatory or inhibitory and organizes them in list and matrix. The matrix of connection and list of neurons generated is then translated to hardware SNN configuration using the available types of excitatory and inhibitory neurons and synapses defined in the existing software, showcasing a case of custom user script to generate a given network structure.

The parameters of the SNN were tuned to match the electrical activity in terms of mean firing, synchronicity and burst activity of each structure obtained from MEA recordings. This was performed by tuning the synaptic noise to obtain a similar spontaneous activity, then improved by modifying the weight of synapses, percentage of connection as well as the inhibition/excitation ratio.

4.3.3 Real-time emulation

The three structures were emulated using 1,024 neurons distributed equally between the two organoids with a similar inhibitory/excitatory ratio to biology (20% inhibitory and 80% excitatory).

Inhibition is modeled using FS neurons connecting by $GABA_AR$ and excitation by RS neurons connecting by AMPAR. The synaptic noise was set to obtain spontaneous activity for neurons of about 1 Hz as observed in biological cultures [Kirihara et al., 2019]. The spiking activity of the network was captured using the on-board saving in file, then linking with the neuron types using the information of the configuration file.

The emulation of the three structures is shown in Figure 4.8 that shows the synaptic connection of the different structures as well as their spiking activity emulated by the system. The emulation time was set to 10 seconds to allow clearer visualization in figures but was emulated for larger periods as shown in the biohybrid experiments detailed in further sections. The benefit of real-time emulation in this specific application corresponds to the time saved compared to software emulation that ran 30 minutes to emulate 5 seconds.

The emulations presented in Figure 4.8 shows the capability to reproduce from network bursts to burst synchronization between organoids in the *assembloid* and *connectoid* structures as shown in Figure 4.8.

The burst synchronization between the organoids is notably demonstrated by comparing the activity of the *assembloid* and *connectoid* with the *single*, where the burst of the organoids are not synchronized in *single* that does not implement synaptic connections between organoids.

The initial burst at the beginning of the emulation can be explained by identical initial parameters for the neurons and synapses that could be fixed by adding a small fluctuations in initial parameters when generating the configuration.

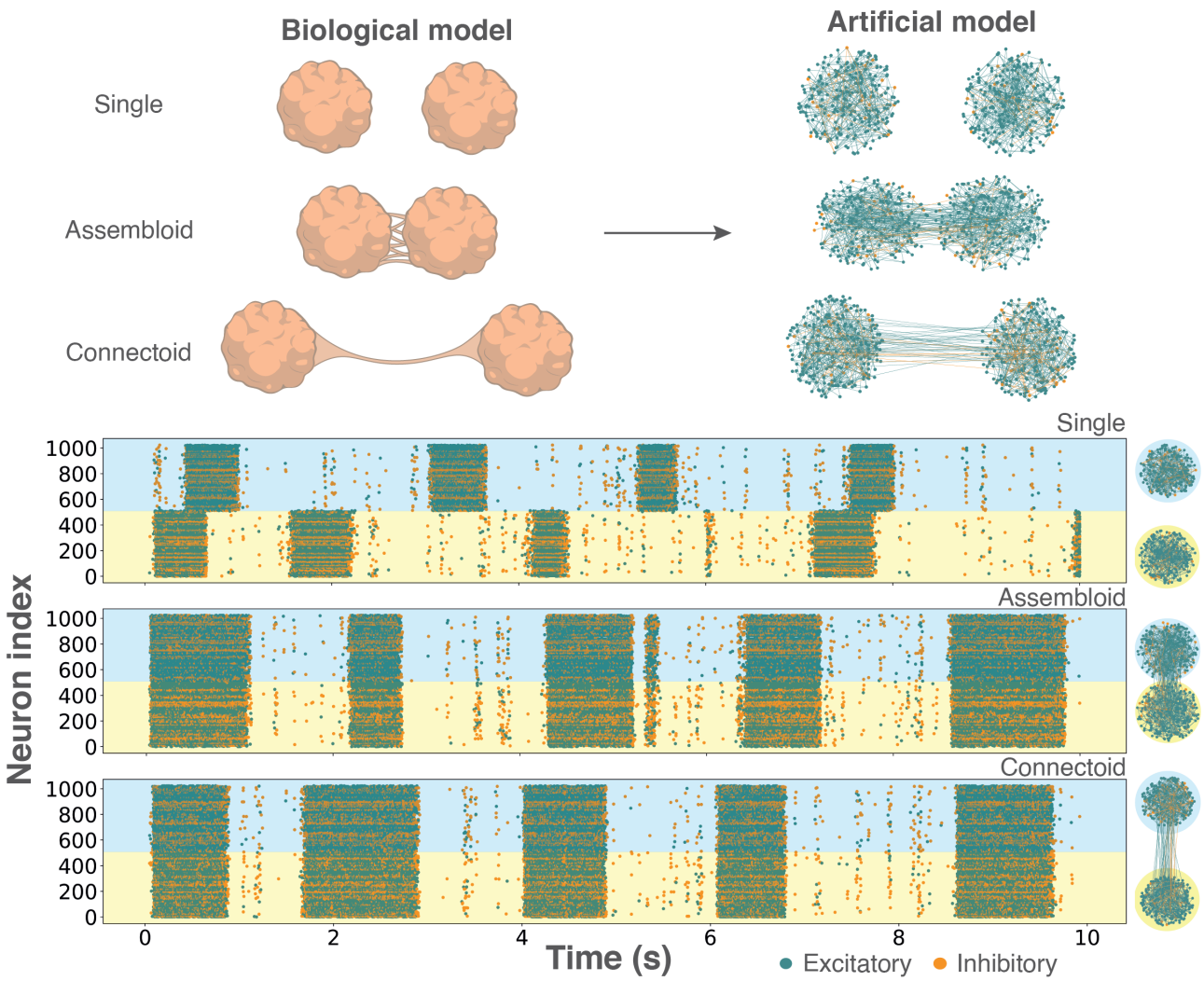


Figure 4.8: Three structures of cortical organoids modeled using FS and RS neurons connected with excitatory and inhibitory synaptic connection (AMPAR and $GABA_AR$) based on biological culture observations and their spiking activity. Synaptic connections are promoted according to rules depending on the structure to reproduce, spatial placement of neurons and the ratio of inhibition/excitation connection observed. The spiking activity emulated corresponds to a 10% maximum probability for each neuron to connect to a neuron inside the organoid and 2% outside. Each organoid is composed of 512 neurons showing a ratio of 20% inhibition/excitatory neuron ratio.

Spiking activity of 5-minute emulations were analyzed for each structure and presented in Figure 4.9. Through the analysis of bursts and spikes, the activity of the single is shown significantly different to the two others interconnected structures.

More specifically, Figures 4.9A,B that shows the MFR and ISI of organoids independently demonstrates the different level of synchronization of the spiking activity.

Where the spiking activity of the two organoids in the *single* are different, the spiking activity of the connectoid and assembloid are much more similar thanks to stronger synaptic connection providing higher synchronicity. Most notably, the connectoid structure that shows the strongest synaptic connection demonstrates the higher synchronicity among the three structures.

The burst analysis presented in Figures 4.9C,D also support the coherence of the model by showing the highest activity for the connectoid structure.

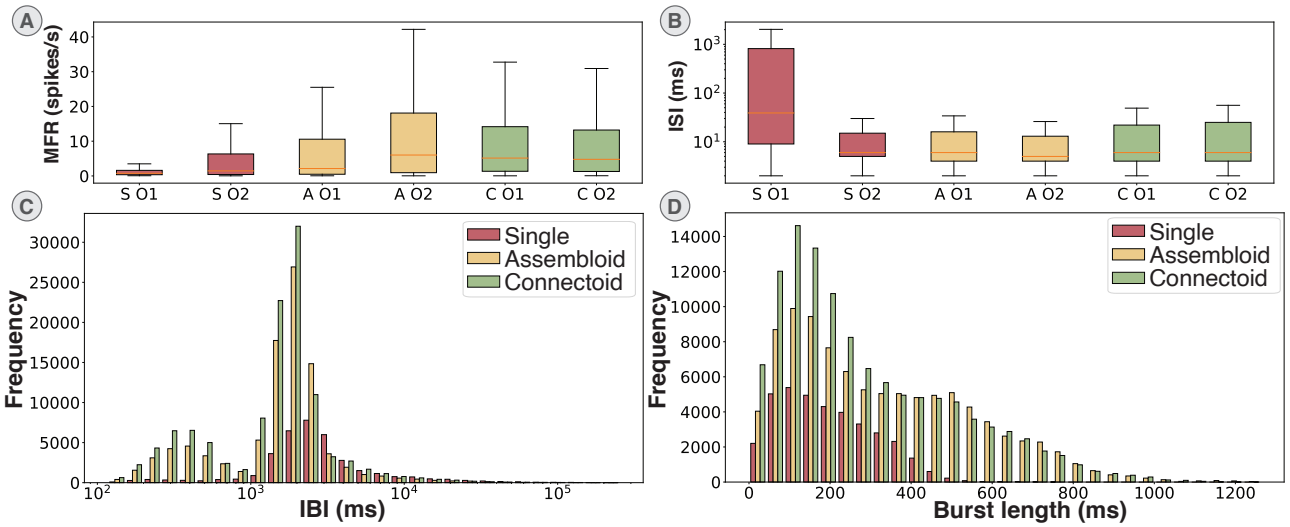


Figure 4.9: Spiking activity analysis of the *single* (S), *assembloid* (A) and *connectoid* (C) structures emulated with BiœmuS for 5 minutes. The maximum synaptic connection probability inside the organoids is set to 10% and to 3% between the organoids. The organoids constituting the structures are abbreviated as O1 and O2. **(A)** Boxplot of the Mean Firing Rate (MFR) for each organoids in the three structures. **(B)** Boxplot of the InterSpike-Interval (ISI) for each organoids in the structures. **(C)** Histogram of the InterBurst-Interval (IBI) in the three structures. **(D)** Histogram of the burst length in the three structures.

The Figure 4.9C highlights a major difference in the bursting dynamics of the structures so as the assembloid and connectoid structures present two modes in the IBI histogram where single only shows one.

The strongest synchronization of the network is also demonstrated by the burst lengths presented in Figure 4.9D that shows longer bursts in assembloid and connectoid structures.

While the organoid modeling demonstrated in this section shows consistent results supported by the spike and burst analysis, the modeling could be improved by introducing more synaptic and neurons types. Additionally, the main drawback of this model is the poor modeling of the dynamics induced by the axon bundle of the connectoid that is responsible for a significant delay not complementary described by the current model.

4.3.4 Mimic drug treatments

An example of complementary application designed using the organoid modeling is the emulation of drug treatments targeting synaptic receptors in the organoid. Two emulations were performed to reproduce a drug treatment by full antagonist of AMPAR (CNQX) and a treatment by full antagonist to $GABA_AR$ (Bicuculine) that deactivate the receptors targeted.

An organoid of similar structure as previously presented is modeled using 1,024 FS and RS neurons connecting with AMPAR and $GABA_AR$ is emulated for 60 seconds in BiœmuS. The inhibition/excitation ratio is set to 20% inhibition and 80% excitation and synaptic connection inside the organoid is set to a maximum of 10%.

During emulation, a trigger is sent to BiœmuS at 20 seconds to disable a given receptor thus mimicking the drug treatment by full antagonist and a second trigger is sent at 40 seconds to reactivate the receptor (see Figure 4.10). The trigger is sent using the same slot as the external stimulation (Ethernet *via* ZeroMQ) that has been modified to allow synaptic deactivation.

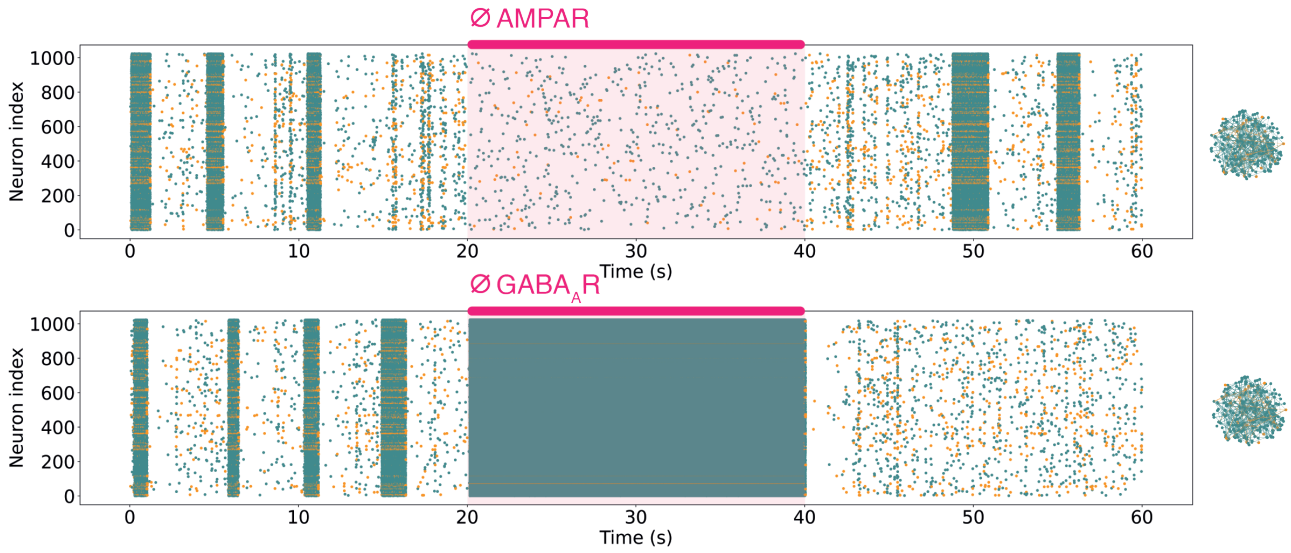


Figure 4.10: Emulation of drug treatment in a single organoid through AMPAR and $GABA_AR$ full antagonists from 20 seconds to 40 seconds.

The results of the emulation shown in Figure 4.10 demonstrate coherent results in both cases. In the case full antagonist to AMPAR introduction, the excitatory synaptic receptors are blocked so as the bursting activity is prevented and activity is desynchronized. For the full antagonist to $GABA_AR$, the inhibitory synaptic receptors are blocked so as the network is continuously spiking activity similarly to an epilepsy, a reaction often observed after introduction of bicuculine [Meldrum and Nilsson, 1976].

4.4 Open-loop biomimetic in-vivo stimulation

A simple case of interaction with the living is to perform a unidirectional stimulation, or open-loop stimulation, from the Artificial Neural Network (ANN) to the Biological Neural Network (BNN). This open-loop stimulation was applied to rat brains as a neuromorphic-based open-loop set-up for neuroprosthetic applications targeting post-stroke rehabilitation studies through electroceutic therapy [Panuccio et al., 2018, Semprini et al., 2018]. This section presents the setup, protocol and results of the open-loop in-vivo stimulation driven by the SNN of BiœmuS performed. The Figure 4.11 illustrates the biohybrid conducted in collaboration with the research team from the University of Genoa and that has been published in [Di Florio et al., 2023].

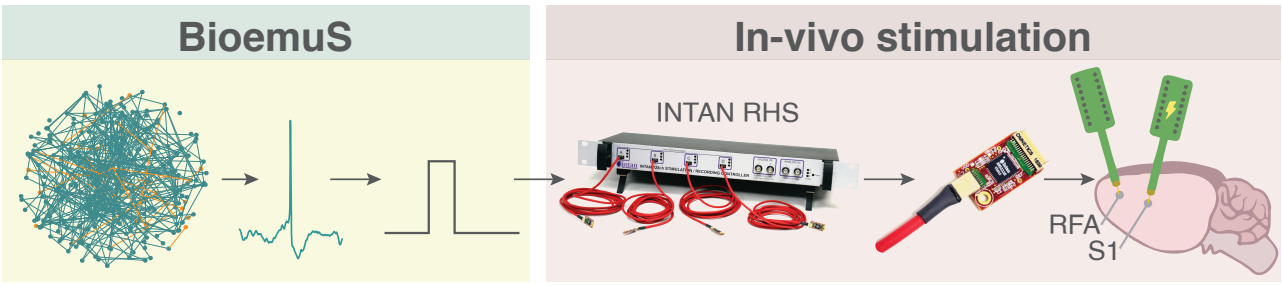


Figure 4.11: In-vivo stimulation driven by BiœmuS spiking activity as a model of post stroke rehabilitation *via* adaptive stimulation. The spiking activity of the SNN triggers stimulation in-vivo using the INTAN RHS2116 headstage. Electrode arrays were placed in the rostral forelimb area (RFA) and in the primary somatosensory area (S1) in the brain of adult rats.

4.4.1 Electroceutical approach for post-stroke rehabilitation

As introduced in the first chapter, neurological disorders are a great burden and especially stroke that represents one of the leading causes of long-term disability and death worldwide.

About 87% of stroke cases are ischemic strokes [Di Florio et al., 2023]. Ischemic strokes are caused by inadequate oxygen and nutrients to the brain tissue that causes rapid structural damage leading to sensorimotor and cognitive impairment.

Generally happening in the primary motor cortex, ischemic events induce a progressive loss of information transmission to the spinal cord, thus causing motor dysfunctions.

Hence, leakage in communication throughout sensorimotor regions (e.g. primary somatosensory cortex (S1)) is noticed to contribute to the severity of symptoms [Carè et al., 2022].

Similarly to neurodegenerative diseases, treatments are limited. Physical therapy is a standard treatment to recover sensorimotor functions, however it often shows limited or incomplete efficacy to promote spared regions reorganization [Chiappalone and Semprini, 2022].

Electroceutical therapy through open-loop and closed-loop electrical stimulation [Averna et al., 2020] thus appears as a promising treatment, especially activity-dependent stimulation (ADS).

Activity-dependent stimulation consists in the detection of action potentials in a region to apply a stimulation in a different region following the principle of Hebbian plasticity [Guggenmos et al., 2013, Jackson et al., 2006].

This biohybrid experiment constitutes a preliminary study of electroceutical therapy to restore sensory motor function in post-stroke rehabilitation, thus constituting a step toward neuromorphic-driven stimulation.

4.4.2 Intermediate version of BiœmuS

The version of the system used in this section corresponds to a former version of BiœmuS that explore a different system integration, architecture and target.

In this version, the target was the ZyboZ7-20 that features the Zynq architecture. Briefly, the target still has a Zynq architecture that includes PS and PL parts, but with overall lower performances. Notably, no commercial Linux distribution supports this architecture so as only a less generic Linux provided by AMD Xilinx is available. The bare-metal approach is then used in this version of the system for faster development.

The main data communication protocol used in the system is the USB2.0 in Communication Class Device (CCD) with the board as the device and a computer as a host. It is implemented in the PS part of the ZyboZ7-20 in bare-metal using the sources provided by AMD Xilinx and using Python on the host computer.

Contrary to the current version that uses a configuration file, the configuration is directly set in the Python application and sent over USB for the C++ application on the target to set the SNN. The spikes were monitored by polling constantly AXI-Lite registers storing the states of all neurons.

Debug information that include status and errors is sent by the target using UART configured at 115,200 bauds.

The Figure 4.12 presents the global system architecture of this version of the system.

The hardware design is based on the work [Khoyratee et al., 2019] so as independent neurons using exclusively fixed point coding and fitted equations for ionic channel states are used.

The parameters of the FS and RS neurons used are the identical to [Khoyratee et al., 2019]. Spikes were considered in hardware when the membrane potential of a neuron crossed 0 mV

and generated a pulse on a 3.3V digital output of the PMODs.

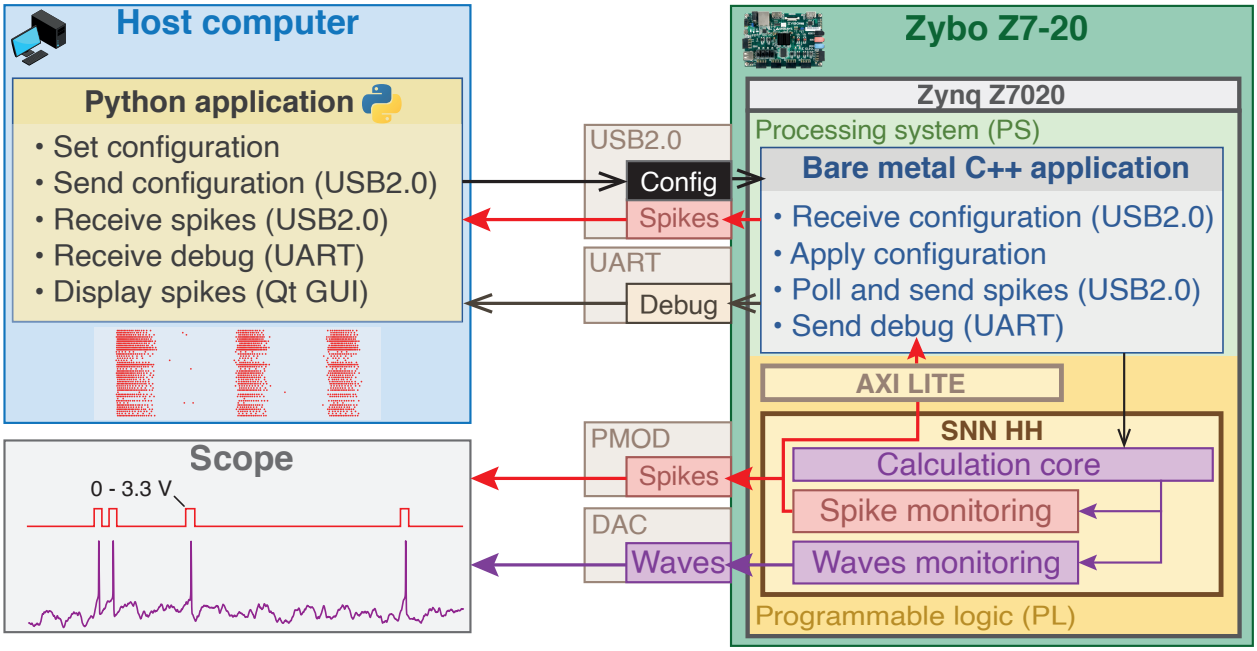


Figure 4.12: Architecture of the intermediate version of BiœmuS using bare-metal C++ application to interact with the system and USB2.0 CDC as the data communication protocol.

4.4.3 Experimental setup and protocol

The animals employed were healthy adult Long-Evans rats. All the rats were treated with the SNN-based stimulation while they were deeply anesthetized. The experimental procedures were performed with the collaborating team in the Animal Facility of the Italian Institute of Technology (IIT), Genoa, Italy and were previously approved by the Italian Ministry of Health and Animal Care (Italy: authorization n. 509/2020-PR).

After a surgical procedure, sharp electrodes were inserted in the primary somatosensory area (S1) and rostral forelimb area (RFA) (see Figure 4.11). The MicroElectrode Array (MEA) inserted were NeuroNexus probes (A4x4-5 mm-100-125-703-A16). The detailed surgical procedure is detailed in [Di Florio et al., 2023].

The acquisition and stimulation from and to the MEA were performed through the INTAN RHS2116 head stage, a bidirectional electrophysiology interface system consisting of 16 or 32 stimulation/amplifier channels connected to the electrodes. The headstage communicates using SPI protocol to communicate with the Intan Technologies RHS Stim/Recording controller, an FPGA-based electrophysiology data acquisition system (see Figure 4.11).

The spikes from neurons emulated by BiœmuS are output as 0-3.3V pulses connected to the INTAN RHS recording/stimulation unit to trigger stimulation upon spike reception.

The spontaneous activity of the Fast Spiking (FS) and Regular Spiking (RS) neurons emulated are tuned to obtain slow and fast activities between 1 Hz and 10 Hz by tuning the parameters the synaptic noise. In this setup, the latency between spike detection and stimulation is less than a millisecond as the INTAN stimulation unit is FPGA based.

The stimulation electrode was chosen for the lower impedance found in channels in S1 based on impedance analysis. The stimulation applied by the headstage was a single biphasic pulse of 60 µA of 200 µs positive and 200 µs negative [Averna et al., 2020].

The experimental protocol consisted of 20 minutes of pre-stimulation (PreS) recording, 60 minutes of stimulation via SNN-driven stimulation and 20 minutes of post-stimulation (PoS) recording. Extracellular signals were continuously sampled at 25 kHz on 16 bits and stored on the computer connected to the INTAN recording unit.

4.4.4 Results

The data were analyzed offline by the collaborating team using a custom Matlab script performing band-pass filtering as in [Carè et al., 2022], followed by spike detection employing precision timing spike detection (PTSD) algorithm [Maccione et al., 2009].

The neuronal activity was evaluated by computing the Mean Firing Rate (MFR) (spikes/s) in the PreS and PoS recordings for all animals as presented in Figure 4.13.

Statistical analysis were also performed in Matlab using non-parametric tests as data failed the Kolmogorov-Smirnov normality test. The Mann-Whitney U-test was performed to identify differences in the MFR on pre- and post-stimulation data and P-values < 0.05 were considered statistically significant. The Mann-Whitney notably allowing to verify the hypothesis that two groups are independent of each other.

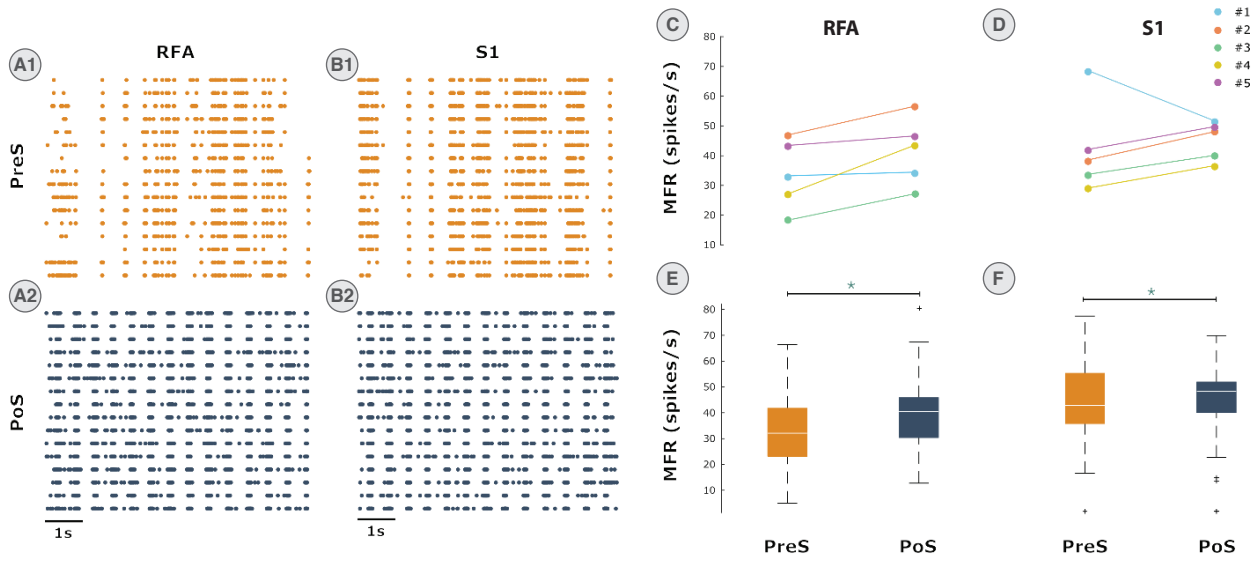


Figure 4.13: Effect of biomimetic open-loop stimulation in RFA and S1. The statistical analysis has been performed considering each channel singularly. **(A1,B1)**: Raster plot of recorded activity in RFA and S1 during pre-stimulation phase. **(A2,B2)**: Raster plot of recorded activity in RFA and S1 during the post-stimulation phase. **(C, D)**: Comparison of the mean firing rate (MFR, spikes/s) between pre- and post-stimulation in RFA and S1 by animals. **(E, F)**: Boxplot of the mean firing rate (MFR, spikes/s) by animals for RFA and S1 regions. The entire dataset has been analyzed with no discrimination among animals to obtain an overall understanding of the stimulation efficacy. Solid line *: p<5.10-2, Mann-Whitney U-test.

From the analysis of the spiking activity observed both in RFA and S1 for the PreS and PoS

phases, the potential variations in the MFR introduced by the SNN-based stimulation allow to evaluate its efficacy to induce an increase of the spiking activity.

The raster plot of a representative animal is reported for both RFA (Figure 4.13A1,A2) and S1 (Figure 4.13B1,B2).

The firing rate appears less synchronous and more intense in the post stimulation compared to the pre stimulation condition. Moreover, the MFR increases in post stimulation phases for all animals except for #1 in S1 area (Figures 4.13C,D).

The quantitative analysis (Figures 4.13E,F) confirmed the observations qualitatively by means of the raster plots.

Specifically, the stimulation induced a significant rise of the MFR from PreS to the PoS phase in both RFA (32.12 spikes/s for PreS against 40.49 spikes/s in PoS, $*p < 9 \times 10^{-4}$) and S1 (44.09 spikes/s for PreS against 50.42 spikes/s for PoS, $*p < 2 \times 10^{-2}$).

To conclude, this preliminary experience shown that stimulation pattern generated by means of the biomimetic SNN shows efficiency to increase the firing activity of both RFA and S1 compared to the pre-stimulation condition.

4.4.5 Discussion

The results obtained are consistent with the previous findings of the collaborating team that showed that the absence of stimulation does not affect the level of firing in both healthy and lesioned animals [Carè et al., 2022, Averna et al., 2020] as well as the effectiveness of a closed-loop stimulation (i.e. ADS) in increasing the level of firing.

These results support the hypothesis of the collaborating team, which suggests that a neural biomimetic pattern more effectively entrains the network in response to stereotyped stimulation, making the population tend to be more responsive to incoming electrical stimuli. This is in line with recent human studies [Cottone et al., 2018].

Hence, the next step would to drive the stimulation from a complete network instead of a single stochastic neuron, constituting a step toward the realization of neuroprostheses and promoting the use of BiœmuS as a tool to investigate stroke rehabilitation in an electroceutic approach by providing activity-dependent stimulation.

4.5 Closed-loop biohybrid spinal cord-brain interaction

Another application designed explores the real-time emulation of a smaller network targeting a smaller embedded target controlling a robot and its interaction with the living. The hardware implemented corresponds to a reduced version of BiœmuS modeling a small network that reproduces a Central Pattern Generator (CPG), an essential network responsible for locomotion found in the spinal cord and characterized by alternating bursts [Marder and Bucher, 2001, Brown, 1914]. This artificial network is then interfaced with a biological neural network embodied by cerebral organoids in a closed-loop fashion to study the biohybrid interactions as a step toward neuroprostheses.

This section presents the artificial modeling of the CPG and its implementation on a robot to mimic snake motion, the details of this different version of the system and finally shows a prototype experiment of biohybrid closed-loop.

4.5.1 Snake robot controlled by biomimetic Central Pattern Generators

Locomotion is one of the most basic abilities in animals that is known to be created by Central Pattern Generator (CPG) activity as observed in swimming for salamanders [Ijspeert et al.,

2007] and lamprey [Cohen et al., 1992]. The activity of CPGs is characterized by alternating bursts [Brown, 1914] that are capable of producing rhythmic patterned outputs without sensorial input.

In the realm of robotics, CPGs are generally made from simple neuron models [Amari, 1972] or simple oscillators [Van Der Pol and Van Der Mark, 1928] that does not operate on biological timescale, thus constituting bio-inspired systems rather than biomimetic systems. In order to provide an artificial CPG capable of reproducing biomimetic CPGs, the team developed a digital neuromorphic system using the Izhikevich model [Blanchard et al., 2019]. The CPG were derived from the neural network controlling the heartbeat of leeches [Hill et al., 2001] and was tuned to obtain a delay between the CPGs to obtain a snake-alike motion on 8 CPGs distributed over 8 wagons thus having 1 neuron per motor. The robot acts as a visual control of the behavior to ease the identification of dysfunctions in the CPGs.

In this work, the existing Izhikevich model was replaced by a more biologically coherent that is the HH model and added Wi-Fi monitoring. The Figure 4.14 illustrates the system updated and shows the spiking activity of the HH CPG controlling the robot.

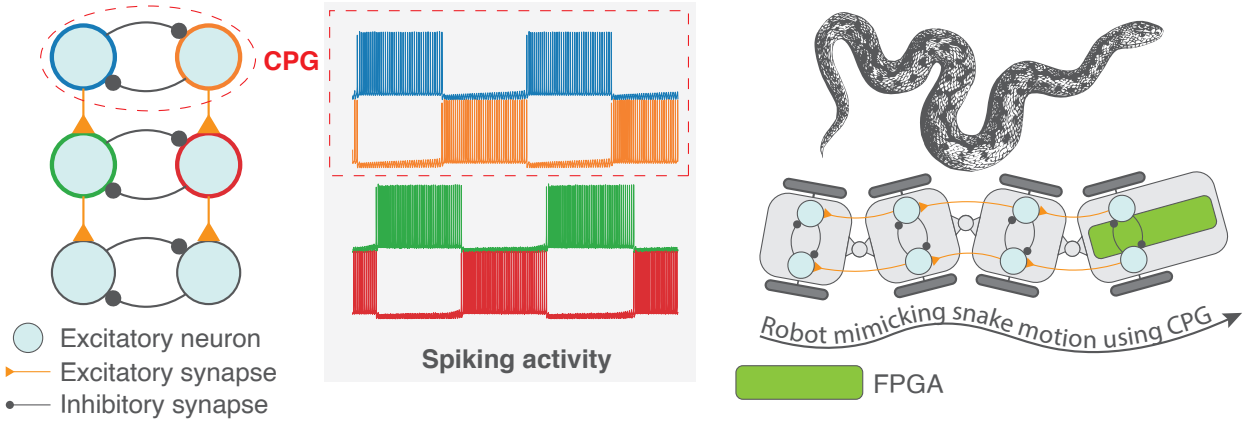


Figure 4.14: Example of reduced version of BiœmuS applied to a smaller target implementing Hodgkin-Huxley (HH) Central Pattern Generator (CPG)s. The small network is reproducing Central Pattern Generator (CPG) network from [Hill et al., 2001] implemented to reproduce snake motion on a robot. CPGs are interconnected by excitatory synapses introducing delay and generate the locomotion. The spiking activity was recorded from the waves monitoring output by the DAC.

Incorporating a more biologically coherent model such as the HH model allows a more accurate modeling of the neural network, hence better suiting the aim of creating an artificial biomimetic spinal cord model. Jointly, it aims to allow investigation the effect of ion conductances through parameters sweeps, ion conductances being usually affected by the neurological disorders.

4.5.2 Alternative version of BiœmuS

The alternative version of BiœmuS presented in this application is based on a previous work of the team [Khoyratee et al., 2019]. This reduced version implements fixed point coding for all operations and fixed configuration of the network set directly in hardware (see Figure 4.15). The equations of ion channel states are simplified and fitted to be equated as operations that can be efficiently implemented on FPGA [Khoyratee et al., 2019].

The platform for this reduced version is the Digilent CMOD A7 that integrates a small target incorporating only programmable logic, i.e. FPGA. As the capacity of the FPGA is limited, the clocking frequency is reduced to 25 MHz and the number of neurons is set to 16.

The monitoring channels available in this version allow both spike and waves monitoring as shown in Figure 4.15. The waves are not necessarily used in this application presented, but they can be visualized using a DAC similarly to the main version. The spikes can be monitored either from the GPIO that controls the motors of the robot or through Wi-Fi *via* the PMOD ESP32. The visualization of the spiking activity over Wi-Fi is performed by Python scripts using Qt-based GUI running on a computer as shown in Figure 4.15.

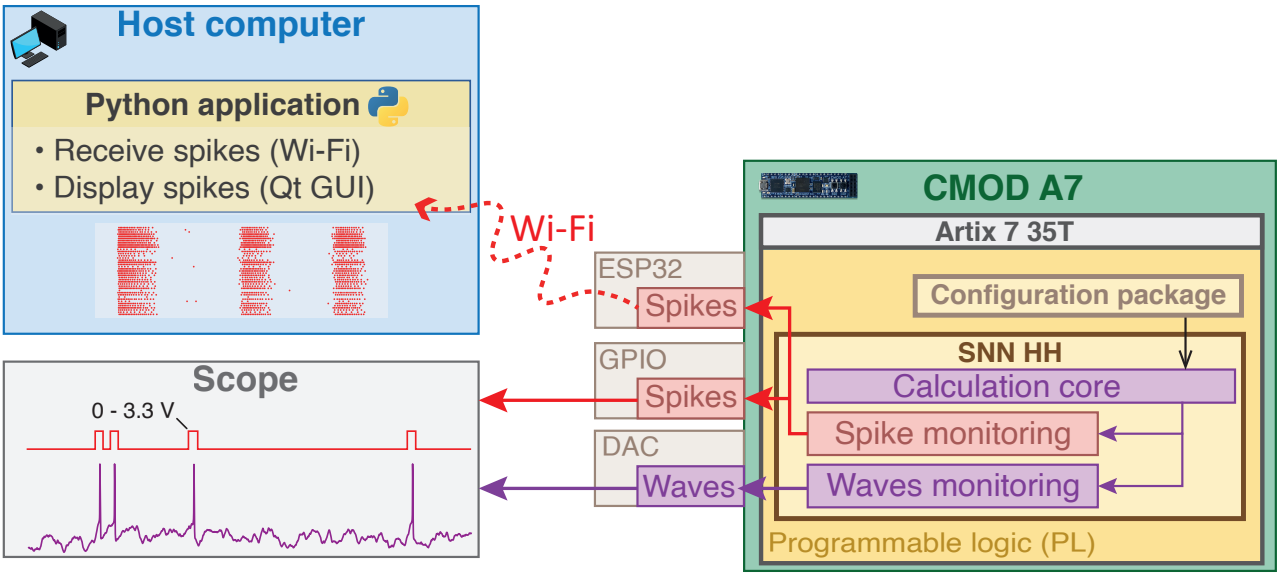


Figure 4.15: Architecture of the alternative version of BiœmuS using only programmable logic on a smaller target communicating through Wi-Fi.

In this version, the power consumption of this system is significantly lower as the target is smaller, operating at a lower frequency and with smaller processors as the ESP32 is designed to allow small power consumption.

To put if briefly, this solution proposes a version suited for embedded applications where the energy efficiency as well as physical constraints are a concern. The main drawback is notably the lack of flexibility and the limited performances.

4.5.3 Experimental setup and protocol

As a preliminary experiment to the development of neuroprostheses, a biohybrid experiment in a closed-loop fashion was conducted to explore the interaction between an artificial model of spinal cord embodied by the snake robot and human cerebral organoids as a brain model. In other words, it aims to mimic artificially the interaction between the brain and spinal cord. This prototype experiment was conducted once as a proof of feasibility and in collaboration with the team from the University of Tokyo. The experimental setup is illustrated in Figure 4.16.

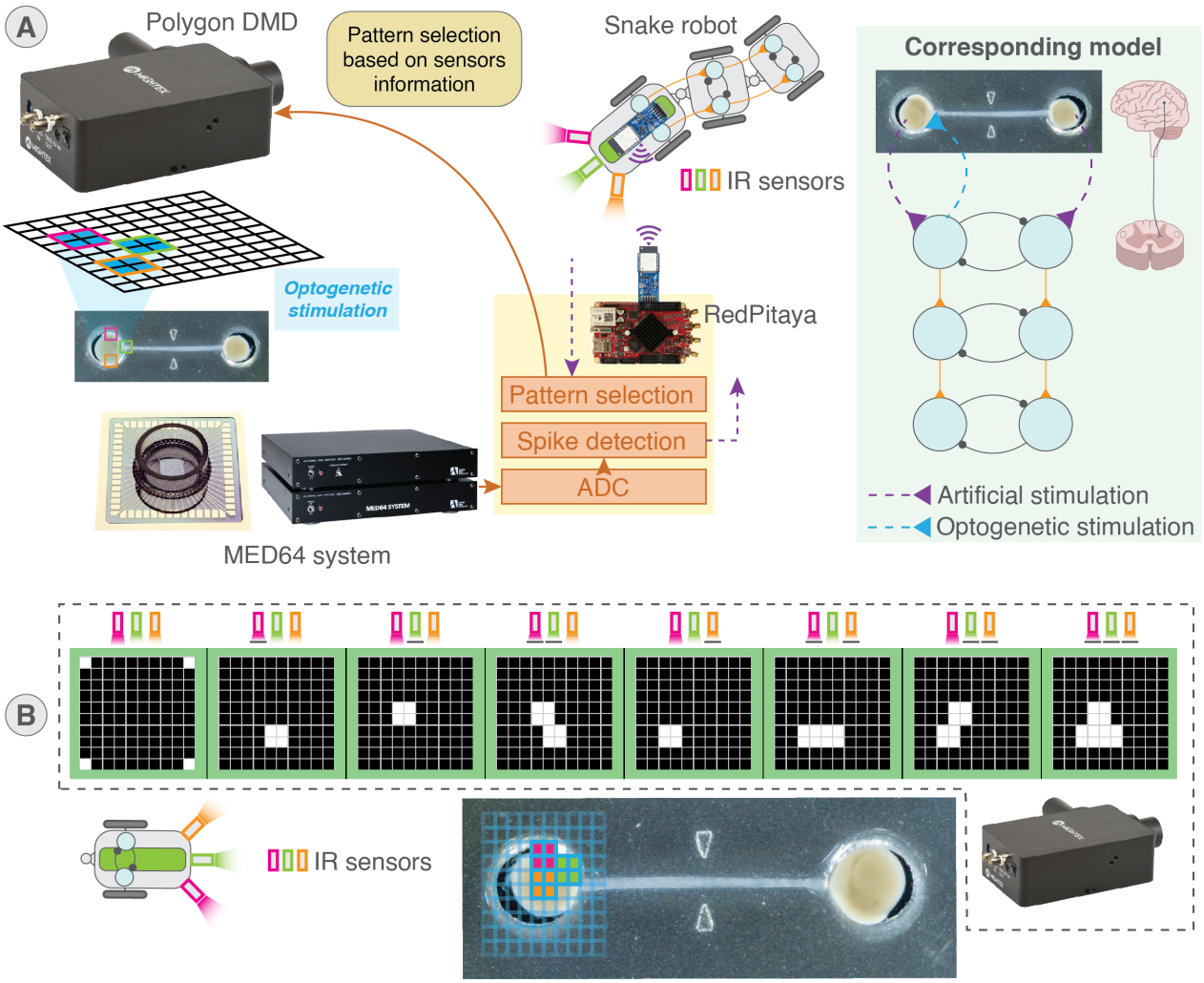


Figure 4.16: Closed-loop biohybrid experiment integrating an embedded version of BiœmuS. **(A)** Closed loop interaction between biological connected cortical organoids and artificial CPG as a model of brain to spinal cord interaction. Spiking activity detected on the left and right organoids is forwarded through Wi-Fi to the first neurons of the artificial CPG implemented on the robot shown from [Blanchard et al., 2019]. Three Infrared (IR) sensors located at the front of the robot (purple, green and orange rectangles) trigger localized optogenetic stimulation on the left organoids using Digital Micromirror Device (DMD) based on the data transmitted by Wi-Fi. **(B)** Sequence of DMD patterns selected based on information of the infrared sensors. The pattern were switched using the trigger in input of the DMD Polygon1000 that moves to next pattern upon reception of a pulse. The pattern was projected on the left organoid.

Starting from the biological network, the spiking activity from left and right organoids stimulates the first CPG of the snake robot.

MEA signals are recorded by the AlphaMED64 system and forwarded to an FPGA (Red-Pitaya STEMlab125-1) that performed digitalization and threshold based spike detection.

Spiking information is then sent through Wi-Fi to the snake robot. The CPG is configured on the snake robot [Blanchard et al., 2019] to provide a stimulation based on the spike activity of the biological culture received *via* the Wi-Fi module (PMOD ESP32).

Sensory feedback is introduced through optogenetic stimulation of the left organoid triggered accordingly to the infrared sensors present in the front of the robot. The information of each sensor is mapped as a square area for each sensor as illustrated in Figure 4.16B.

The optogenetic stimulation is performed using a Digital Micromirror Device (DMD) and

made possible by a genetic alteration of the organoids. The sequence of illumination patterns was designed using Polygon software provided by the manufacturer of the DMD.

On reception of the information of sensors, the RedPitaya selects the corresponding pattern on the DMD. The selection of the pattern is done using the trigger in input that allows to switch patterns from pulses (see Figure 4.16A). Hence, the appropriate number of pulses were sent based on IR sensor information to select the matching pattern.

The cortical organoids used in this experiment were cultivated using the procedure described in [Kirihara et al., 2019, Kawada et al., 2017].

The connected organoids were plated on MEA and infected with Channelrhodopsin-2 to allow optogenetic stimulation. More specifically, 1 µl of AAV-CAG-hCHR2-tdTomato (Vector-Builder Inc.) was added to the culture media at day 60 experiment for optogenetic interventions as in [Osaki and Ikeuchi, 2021]. The infection rate was confirmed through imaging at day 100 as depicted in Figure 4.17A.

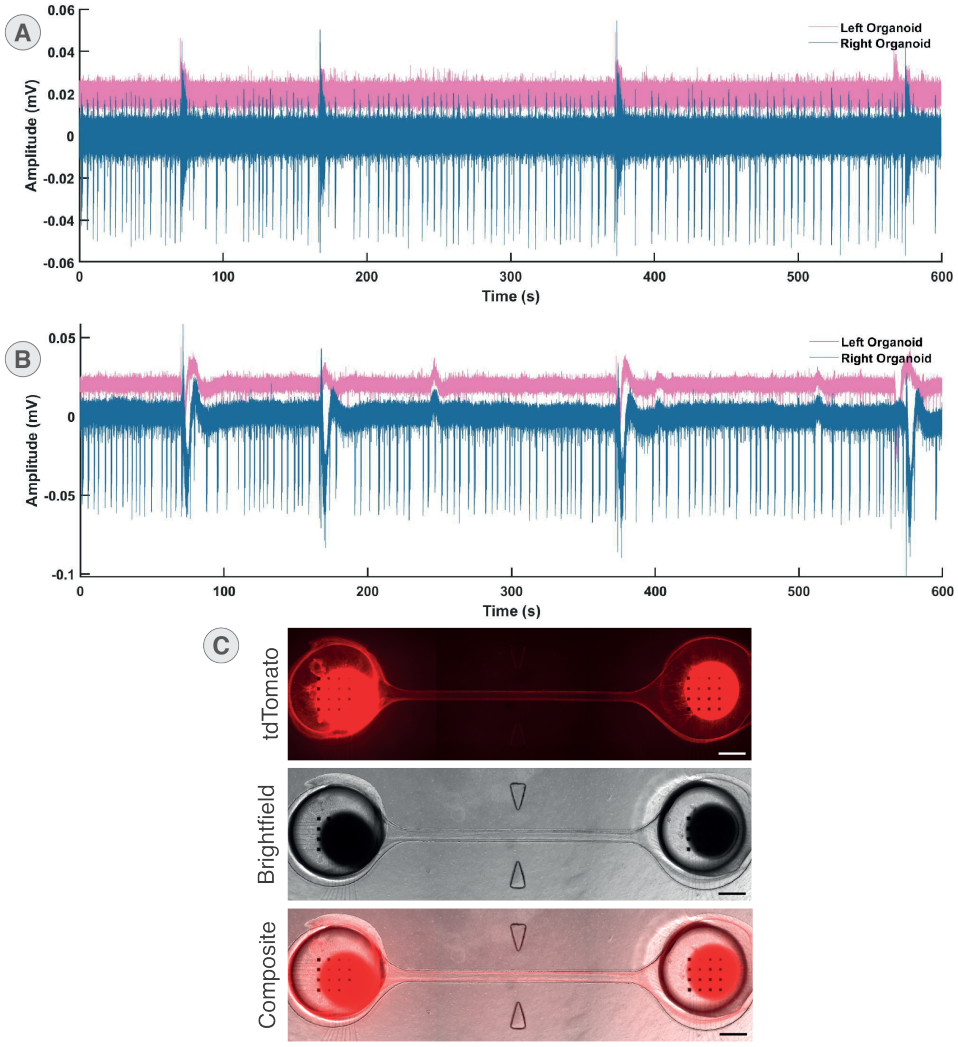


Figure 4.17: Connected human cerebral organoids used in the closed-loop biohybrid experiment. **(A)** Band pass filtered electrical activity of left and right organoids on day 92 recording on MEA (low-cut 300 Hz, high-cut 3 kHz). **(B)** Raw electrical activity of left and right organoids on day 92 recording on MEA (low-cut 300 Hz, high-cut 3 kHz). **(C)** Connectoid organoids expression of ChR2 imaged at day 100. Infection and plating at day 60. Scale bar is 500 µm.

The activity of the organoids was recorded using the AlphaMED64 system and checked regularly to ensure that the cultures were still alive, active and shown spiking activity as presented at day 92 in Figure 4.17B. The experiment was performed at day 110.

The feasibility of the system was demonstrated qualitatively by running the system for 10 minutes and filming the behavior of the robot as well as visualizing the stimulation triggers of the DMD and spiking activity of the cultures.

4.5.4 Results

The Figure 4.18 presents shots extracted from the video of the experiment that shows an example of optogenetic stimulation.

As the robot gets closer to the obstacle, the IR sensor is set high and select the appropriate pattern to illuminate on the culture (Frames 2,3 in Figure 4.18). When the robot moves away, the sensors are low and the associated pattern is selected (Frames 4,5 in Figure 4.18). While the experiment certainly cannot prove that the robot dodges the obstacle, it can be confirmed that the spiking activity of the CPG is modified by the spiking activity of the culture. Indeed, the robot does not move following a straight line in a snake-like motion but rather moves in a disorganized way because of the spiking activity of the culture.

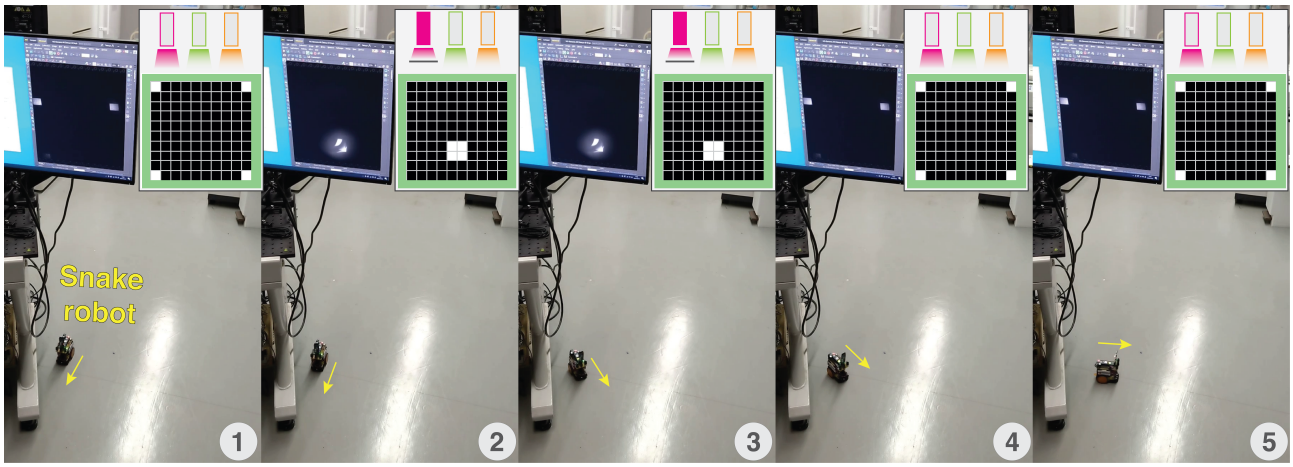


Figure 4.18: Images extracted from the video of the biohybrid experiment conducted showing the trigger of the DMD stimulation based on IR sensors. The illumination of the corresponding pattern on the DMD is selected based on the information of the infrared sensors. The illuminated pattern is visualized on the interface provided by the software controlling the DMD. Only the first wagon was used in this experiment to facilitate the movement as space was limited.

In this setup, the latency between spike detection from the organoids and stimulation on the snake robot is between 5-10 ms and about the same from the CPG to the organoids that corresponds to Wi-Fi latency.

The power consumption as well as the resource utilization of this version are significantly lower than for the main versions as shown in Figures 4.19A,B.

The difference in power consumption is explained by the operating clock frequency, the resources used by the calculation core as well as the PS part in the case of the main version.

As for the resource utilization, the difference is explained by the higher flexibility and performances of the main version, especially the floating point coding and the fully connected flexible synaptic connections.

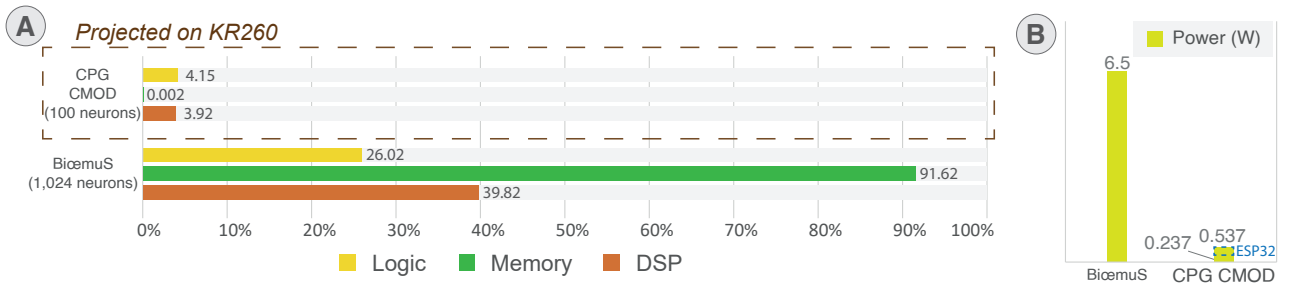


Figure 4.19: Comparison of the resource utilization and power consumption between the main and alternative versions of BicœmuS for the calculation core only. **(A)** Resource utilization projected on the target KR260. The calculation core includes the synapses that are hardware locked for the CPG and flexible in the main version. **(B)** Power consumption on the respective targets. The power consumption of the ESP32 was estimated here as the peak consumption of 300 mW.

4.5.5 Discussion

Although promising, the experiment conducted shows multiple limitations as it was thought as a prototype. Indeed, it focuses more on the feasibility than on the reliability so as validations were kept to a minimum.

The next step would be to conduct complex validations of the system to estimate the reliability of the experiment and to design a protocol that would allow to assess the efficacy of the closed-loop.

Nonetheless, this complex experiment requires a considerable amount of preparation to cultivate the organoids as well as to prepare the setup. Additionally, cross-disciplinary skills are mandatory to ensure the good operation of the experiments as electronics and biology are deeply intricate. Moreover, the effects of the electrical stimulation on human cerebral organoids is not yet completely identified [Osaki and Ikeuchi, 2021], thus requiring additional studies.

To conclude, this experiment constitutes a preliminary work toward the realization of neuroprostheses and points out the challenges it involves. While a significant amount of points needs to be validated and clarified, it has been demonstrated that biohybrid closed-loop interaction remains feasible.

4.6 Closed-loop biomimetic in-vitro stimulation on high resolution MEA

In line with the biohybrid experiments exploring hybridization in the context of the realization of neuroprostheses, this experiment shows integration with an existing biophysical acquisition system in a closed-loop fashion. Additionally, this application promotes the ease of integration of the system with existing solutions for biological interfacing as well as its versatility, here integrated with the new generation of High Density MicroElectrode Array (HD-MEA)[Ballini et al., 2014]. This section introduces the purpose of the experiment, presents the experimental setup. Then, the results obtained are presented and discussed.

4.6.1 Toward biological intelligence using cortical connectoid

Cortical connectoids are neural circuit tissues made of cortical organoids connected together, focusing on the importance of connections between regions in the brain for brain function [Kirihara et al., 2019, Osaki and Ikeuchi, 2021]. Connectoids are active and complex structures that respond to external stimuli.

They constitute the main focus of our collaborating team at the University of Tokyo that is aiming to build an efficient system for training neural tissue, and eventually to make neural tissue spontaneously possess higher-order functions similar to what could be called intelligence.

Therefore, the biohybrid closed-loop constitutes an interesting approach to study the interaction between artificial and biological as well as explore new training system based on biomimetic and adaptive stimulation. As a biologically coherent model, the organoids emulated artificially on BiœmuS constitute a way to deliver biomimetic stimulation.

This would allow investigation of new training methods that could participate in the creation to biological intelligence, in this specific case here organoid intelligence, an emerging and promising field [Smirnova et al., 2023].

More in line with the study of the neurological disorders, as cortical connectoids model the connection between regions of the brain, the interconnection with a biologically coherent artificial neural network that could also mimic different region of the brain would allow predicting and studying the different interactions.

For instance, it would rely on the emulation a model of mid-brain organoid that interacts with a biological cortical organoid to compare to its activity to a completely biological model.

To go further, tuning the parameters of the artificial organoid to mimic the activity of a neurodegenerative disease could help study its impact on healthy cultures. The other way around would allow to investigate the impact of a healthy artificial organoid on an affected culture similarly to the adaptive simulation presented in the experiment of Section 4.4.

4.6.2 Experimental setup and protocol

Connected organoids were plated on HD-MEA. Electrodes were configured to allow activity recording on left and right organoids while allowing stimulation of the right organoid. A single organoid was modeled using BiœmuS on a network of 1,024 neurons and emulating for 180 seconds. Spiking activity of BiœmuS was forwarded to the computer hosting the controlling the HD-MEA system using ZeroMQ over Ethernet and stimulation was sent using ZeroMQ on the external stimulation port of BiœmuS. A Python script executed on that same computer sent stimulation to the HD-MEA upon receipt of a burst from BiœmuS. The Figure 4.20 illustrates the experimental setup described.

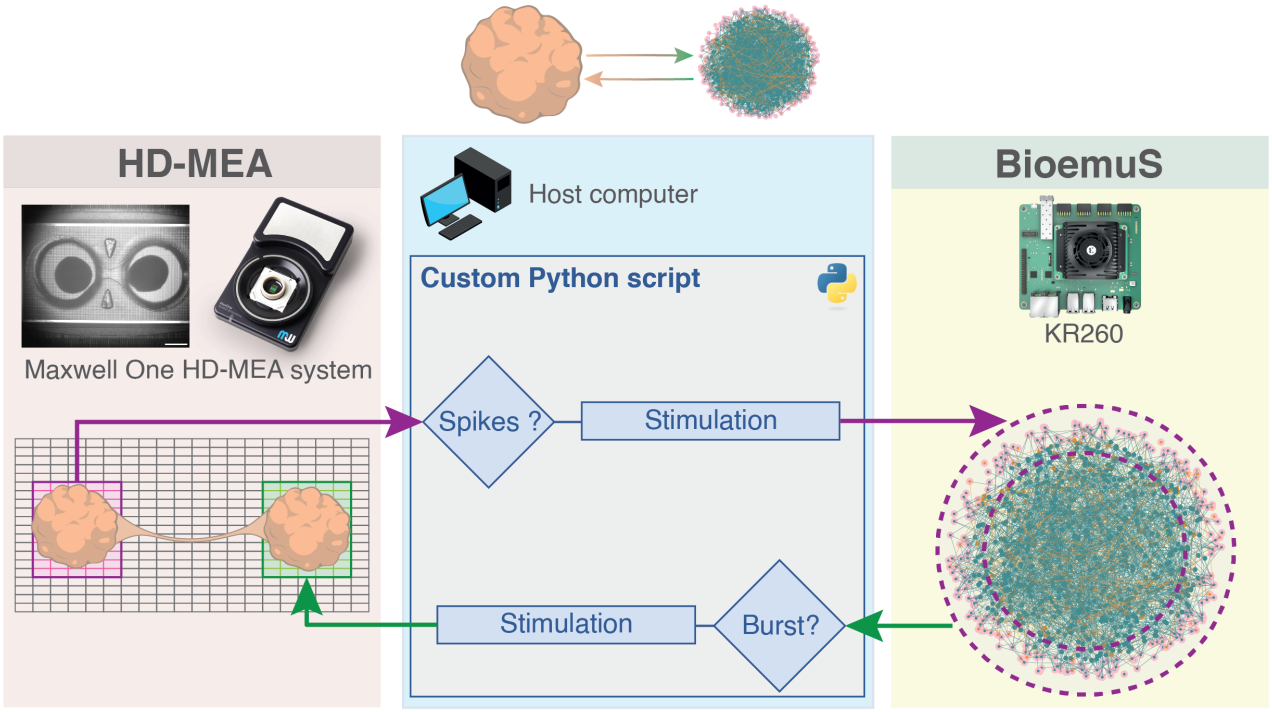


Figure 4.20: Closed-loop interaction between connected organoids plated on High Density MicroElectrode Array (HD-MEA) system and single organoid emulated on BiœmuS. The spiking activity detected in the left organoid of the connectoid in the last 100ms triggers stimulation on exterior neurons of the emulated single organoid on BiœmuS. The bursting activity detected on BiœmuS triggers stimulation on the right organoid of the connectoid. Detection and stimulation commands are carried out by Python scripts using. Stimulation on the SNN is performed using the external stimulation slot.

Biological Neural Network (BNN) on HD-MEA

Starting from the biological neural network, the cortical connectoid was generated using the previously reported protocol [Osaki and Ikeuchi, 2021] and plated on the HD-MEA MaxOne chip of MaxWell Biosystems AG. at d60. This part corresponds to the left part of the Figure 4.20. The Figure 4.21A shows the connectoid on HD-MEA after 2 weeks in device (d84).

The HD-MEA was configured to record from 1,024 channels both from left and right organoid based on an activity scan and to select random stimulation electrodes on the right organoids. The Figure 4.21B shows the electrode configuration selected for the experiment based on the activity scan performed for which analysis is shown in Figures 4.21C,D. The high firing rate observed for the left organoid supported the choice of this latter to trigger the stimulation to BiœmuS as presented in Figure 4.21C. The configuration of the electrodes was extracted to associate the channel numbers to the left and right organoids.

The activity of the biological neural network on the HD-MEA was recording using the MaxLab Live Software started manually before starting BiœmuS. The activity was analyzed using the scripts provided by the manufacturer. The experiment was performed at after 3 weeks in device (d92).

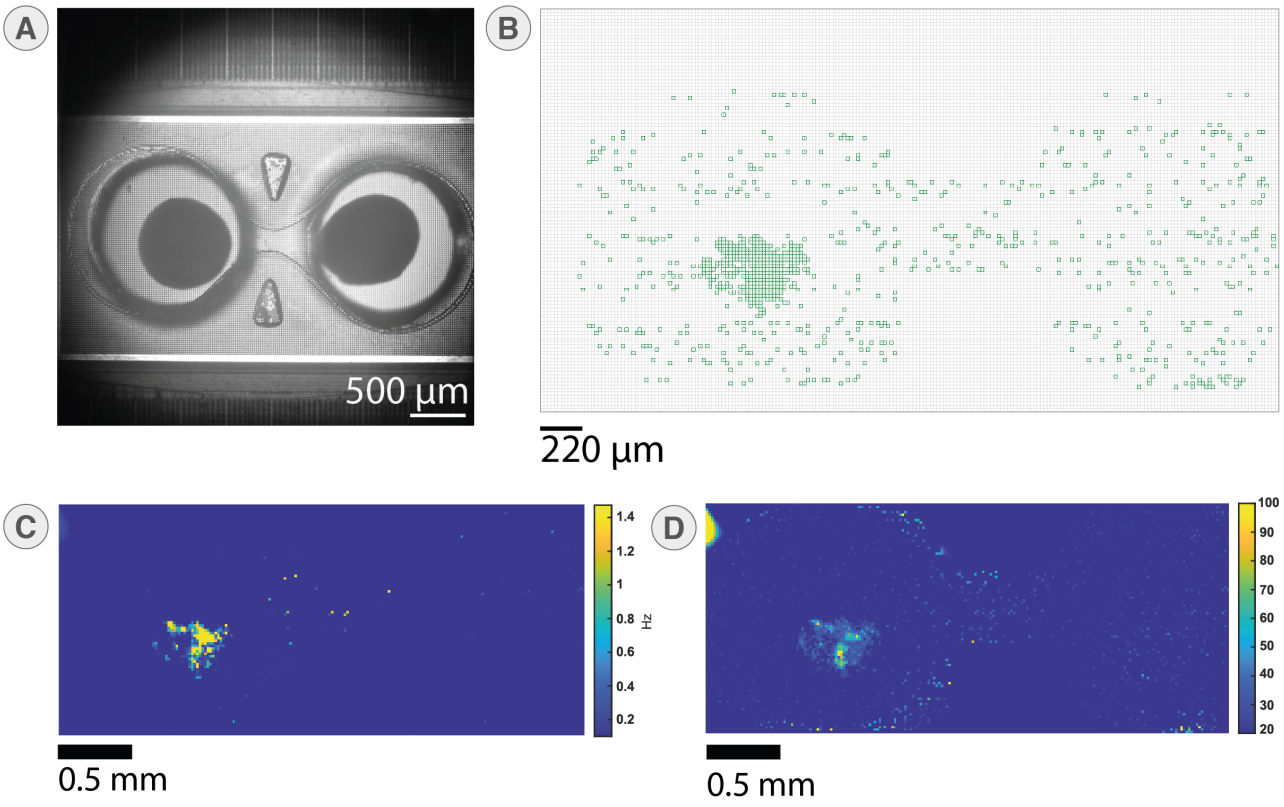


Figure 4.21: Configuration of electrodes of the HD-MEA for closed-loop stimulation on cortical connectoid. **(A)** Connectoid organoids on MaxOne HD-MEA at day 84 in device. **(B)** Electrode configuration of the HD-MEA chip for the experiments showing the 1,024 recording channels including 32 stimulation channels. Electrodes configuration is based on an activity scan carried out with MaxLab Live Software. **(C)** Firing rate map of the connectoid showing higher activity in the left organoid one day before the experiment. **(D)** Spike amplitude map of the connectoid one day before the experiment.

Artificial Neural Network (ANN) on BiœmuS

This part corresponds to right part of Figure 4.20 that corresponds to the Artificial Neural Network (ANN). A single 1,024-neuron organoid was emulated on BiœmuS configured using the Python scripts presented in Section 4.3. Hence, the organoid is constituted of FS and RS neurons connecting with AMPAR and $GABA_aR$. The inhibitory/excitatory ratio was set to 20/80 and the synaptic connection was set to 10 % to fit biological activity. In order to reproduce coherent synaptic connection in a connectoid, only the neurons on the exterior ring of the organoid were receiving external stimulation. This was performed by adding a function that extracted the index of neurons that were located at normalized distance to the center larger than 0.85.

Spiking activity of BiœmuS was forwarded to the computer controlling the HD-MEA system using the ZeroMQ spike monitoring channel over Ethernet. The stimulation of BiœmuS neurons was triggered by the external stimulation port of BiœmuS that uses ZeroMQ over Ethernet. The data collection interval for spikes was set to 100 ms to prioritize reliability over reactivity as lower data collection interval would represent higher loads on both the CPU of the system and the computer that have not been tested yet. The spiking of activity of BiœmuS was recorded on-board in binary format with each bits corresponding to the spiking status of each neuron, thus ensuring a constant amount of data to store.

Interconnection between ANN and BNN

The bidirectional communication between BiœmuS and the HD-MEA system is ensured by Python scripts running on a gateway computer that receive all data and control stimulation triggers. The spikes received from BiœmuS on the host computer are analyzed to detect the presence of a burst in the 100 ms of activity sent. A burst is defined as more than 64 neurons spiking at least 15 times in the last 100 ms of activity received. Upon burst detection, a stimulation corresponding to one period of a 100 Hz sinus wave of amplitude 40 mV is sent to the HD-MEA using custom Python script designed from the scripts provided by the manufacturer. Stimulation was chosen of an amplitude high enough to allow visualization of the stimulation on the MaxLab Live Software.

The spikes received from the HD-MEA triggered stimulation on BiœmuS if at least 1 spike was detected on at least 2 channels in the last 100 ms of activity collected. The stimulation was sent through Ethernet over ZeroMQ to the external stimulation port of BiœmuS to trigger a stimulation of 6.250 ms of 0.03 mA/cm^2 on the neurons on the exterior rings of the organoid to trigger a spike in stimulated neurons.

The Python script implemented a thread for each task of receiving spikes from HD-MEA, receiving spikes from BiœmuS, sending stimulation to HD-MEA and sending stimulation to BiœmuS. Hence, all the tasks could operate concurrently to obtain a consistent bidirectional communication.

4.6.3 Results

As a reference, the spontaneous activity of both networks were recorded the day before the experiment as displayed in Figure 4.22.

The activity of the ANN was configured to have synaptic connections that are not strong enough to spontaneously induce network burst. The Figure 4.22A validates the configuration by observing some synchronized activity but no network burst, except at the initialization as a result of identical initial parameters. The spontaneous spiking activity of the BNN was active with sparse activity as well as synchronized bursts for both the left and right organoids as observed in Figure 4.22B.

Consequently, the requirements in terms of spontaneous activity for the closed-loop experiment were met.

The monitoring of the spiking activity for the ANN and BNN is performed using the MaxLab software for the BNN on HD-MEA that displays the recording activity for each electrode according to its location and marks spike detected with a red triangle (upper part of Figures 4.23A,B,C,D). On the other hand, the spiking activity of BiœmuS is monitored using the Qt GUI (lower part of Figures 4.23A,B,C,D). The Python application responsible for the interconnection of the two systems displays logs in the terminal.

The Figures 4.23A,B,C,D illustrate the functioning of the closed-loop based on the visualization of the activity of the BNN on MaxLab software and of the ANN on the GUI.

In Figure 4.23A, the activity of the BNN on HD-MEA is high enough to send a stimulation trigger to BiœmuS. Once the stimulation is applied on BiœmuS, a network occurs as shown in Figure 4.23B. Upon detection of a network burst from BiœmuS, a stimulation is applied to the HD-MEA as highlighted by the red triangles as it is the case in Figure 4.23C, the red triangles representing here stimulation artifacts rather than spikes. The Figure 4.23D shows the case of a low spiking activity in both networks that does not lead to stimulation triggers.

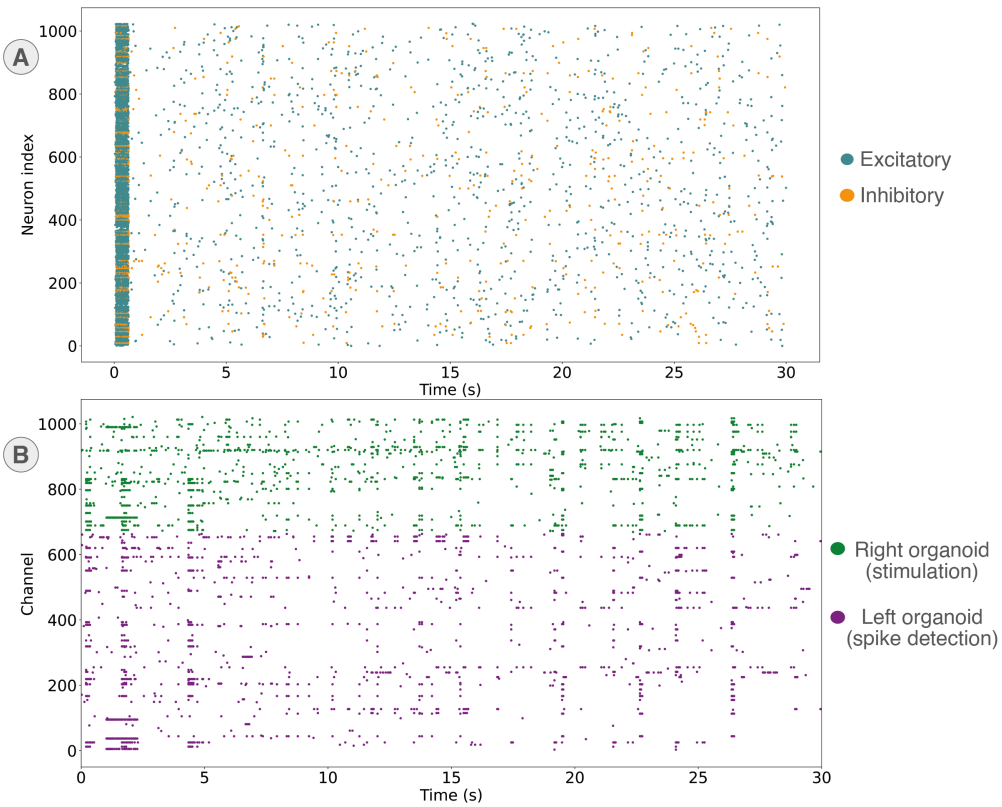


Figure 4.22: Spontaneous activity of the ANN and BNN one day before the experiment. **(A)** Spontaneous activity of the emulated single organoid emulated by BiœmuS. Green dots represent excitatory neurons (RS) and orange dots the inhibitory neurons (FS). Burst occurs only at initialization due to identical initial values for the parameters of the neurons. **(B)** Spontaneous activity of the connectoid on HD-MEA chip one day before the experiment. Green dots correspond to the channels recording from the right organoid that receives the stimulation. Purple dots correspond to the activity of the left organoid that triggers stimulation to BiœmuS based on spiking activity.

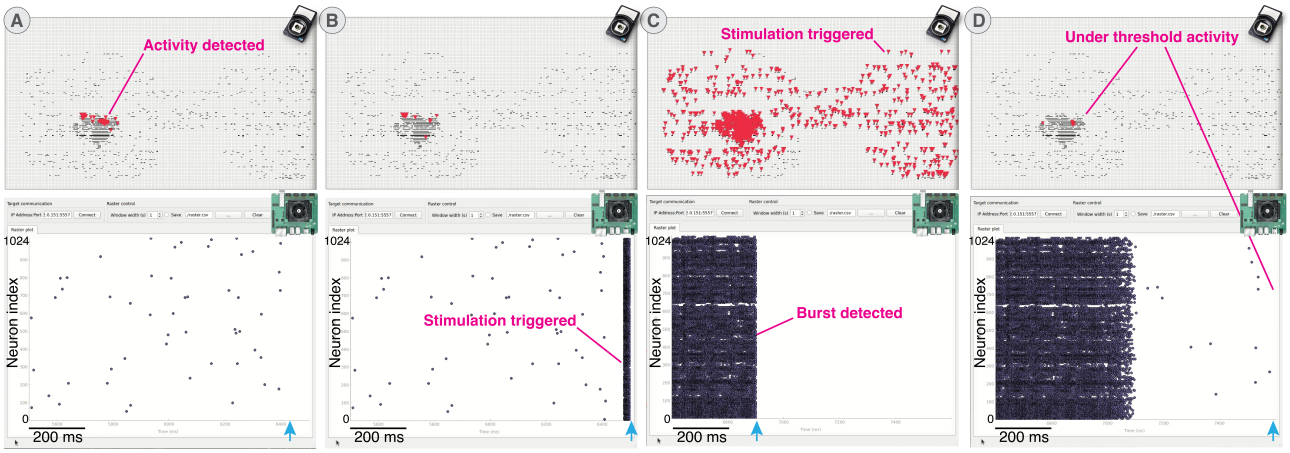


Figure 4.23: Extracted shots from the screen recording of the experiment on the gateway computer showing the monitoring. Spike detected on electrodes of the HD-MEA are shown by red triangles in the MaxLab software. Blue arrows show the current time on the spike monitoring window. **(A)** Sending stimulation to BiœmuS thanks spike detection threshold crossed on HD-MEA (top). **(B)** Stimulation applied to BiœmuS leading to a network burst (bottom). **(C)** Detection of a network burst (top) and stimulation applied to the HD-MEA (top). **(D)** End of stimulation and low spiking activity unable to trigger stimulation.

The Figure 4.24 presents the spiking activity recorded on the HD-MEA and emulated by BiœmuS during the closed-loop experiment. The configuration of electrodes of the HD-MEA was exported from the software. The XY configuration of neurons, network configuration and stimulated neurons of BiœmuS were exported from the Python scripts. Detection of burst and spikes triggering stimulation for both HD-MEA and BiœmuS were reconstructed from the recorded data. The synchronization of both activities was done manually based on the trigger of the first stimulation considering an approximation of 100 to 300 ms based on the latency of the HD-MEA communication and the fluctuating latency induced by the Ubuntu operating system.

The spiking activity of the ANN presented in Figure 4.24 (top) differs from the spontaneous activity previously recorded through the occurrence of network bursts induced by the stimulation from the HD-MEA.

As for the spiking activity of the BNN on the HD-MEA displayed in Figure 4.24 (bottom), stimulation artifacts are interpreted as spikes by the software, thus translating the occurrence of stimulation triggers as "network bursts" on the raster plot. The ANN directly emulating the membrane potential of neurons to detect spikes, the phenomena of stimulation artifacts observed in biological recording is not modeled.

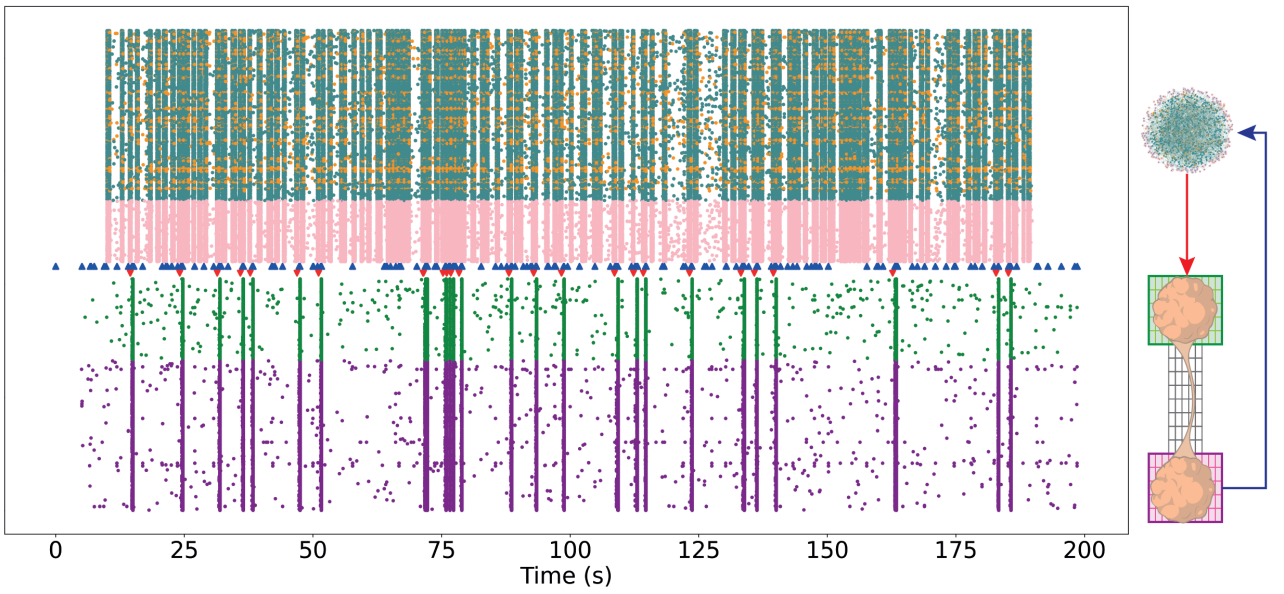


Figure 4.24: Raster plot of the spiking activity for the biological (bottom) and artificial networks (top) for 180 seconds of closed-loop interaction. BiœmuS stimulation triggers are shown by blue triangle and stimulation triggers to HD-MEA by red triangles. BiœmuS is running for 180 seconds starting from 10 seconds and synchronize manually with HD-MEA activity based on the first stimulation trigger ± 300 ms. The activity of BiœmuS was recorded using on-board file saving and the activity of the HD-MEA from the MaxLab software.

While synchronization of the stimulation trigger with spiking activity was set manually for this experiment, it appears that the stimulation triggers and the respective spiking activities are more likely to be correlated according to the definition set in the experimental setup.

Regarding the stimulation trigger applied to HD-MEA (red triangles in Figure 4.24), all triggers are preceded by a network burst.

As for the stimulation triggers to the ANN on BiœmuS (blue triangles in Figure 4.24) that are more numerous, the absence of stimulation triggers always correspond to a period of low activity of the BNN.

In this experiment, the activity of the BNN shows fewer bursts than the spontaneous activity recorded the day before (see Figure 4.22) because of an infection started to occur in the culture and reduced the Spontaneous activity.

This preliminary experiment validated the feasibility of a biohybrid closed-loop integrating the system developed. The spiking activity of the ANN was successfully modified by the stimulation driven by the activity of the BNN on the HD-MEA identified by the occurrence of network bursts. Reciprocally, a stimulation was successfully applied to the BNN depending on the activity of the ANN, confirmed by the appearance of stimulation artifacts after a stimulation trigger following a burst detection from the ANN. While the impact of the electrical stimulation on the BNN is not evaluated here, it is known to have an effect on the biological in-vitro cultures [Levi et al., 2018].

4.6.4 Discussion

While this preliminary experiment most certainly demonstrated the feasibility of an interaction between ANN and BNN, it only validates the concept and enforce the need to conduct further validations and characterization of the system, especially in terms of timing and stimulation effects. Nonetheless, it points out the benefits and challenges of the interaction of BiœmuS with the HD-MEA device.

It emphasized the benefit of the user-defined model through customizable Python scripts to adapt to a specific application, showcased here by the association of XY coordinates to neurons to take advantage of the spatial resolution provided by the HD-MEA and add more coherence to the model. The spatial resolution of the HD-MEA also constitute a considerable benefit in view of an interaction with artificial multicompartment neurons.

The main challenges raised by this experiment with the MaxOne HD-MEA concern essentially the timing, characterization of the stimulation effect and handling of stimulation artifacts.

Indeed, a more reliable tracking and characterization of the latencies involved in the system are required to obtain a more reliable system. As both the HD-MEA through the gateway computer and the BiœmuS are integrating non-real-time operating system, a fluctuating latency is involved. Additionally, the communication and processing latencies are also to be considered as the MaxOne is specified to observe a latency around 100 ms to process data and detect spikes and the BiœmuS monitoring shows limitation on the data collection interval of spikes. Not to mention the manual synchronization that significantly impact the reliability. A viable alternative to manual synchronization would rely on the use of the GPIO featured on the MaxOne chip that would allow capturing triggers along with the recorded data.

Another challenge lies in the processing of the data itself and more specifically in the handling of stimulation artifact. In the experiment conducted, the stimulation artifacts were detected as spikes in the MaxLab software. To obtain a reliable closed-loop, it is essential that the detection of the spiking activity corresponds strictly to spikes and not to stimulation artifacts. Hence, the spike detection of the MaxLab software should be updated to ensure that the activity observed is coherent, not a trivial to perform online.

In line with the challenges associated with the stimulation, electrical stimulation is kwown to have an effect on the culture [Levi et al., 2018], but the actual efficacy of the electrical stimulation for training purposes on organoids remains to be further characterized. As of right now, the effect of electrical stimulation on organoid is not fully understood. Thus, electrical stimulation can not be identified with certainty as the best stimulation method to interact with organoids, thus pushing the consideration of others methods [Smirnova et al., 2023].

To conclude, this experiment presents preliminary results demonstrating the feasibility of a biohybrid closed-loop integrating the system developed and a new generation of standard biophysical interface that is the HD-MEA. This experiment notably showcases the potential of BiœmuS to operate as a tool to study the impact of adaptive stimulation on a culture following the principles of electroceutics while highlighting its ability to adapt to a standard biophysical interface. In the end, this experiment also tackles essential challenges also found in the realization of neuroprostheses.

4.7 Summary

This section presented applications and experiments conducted with the system developed showcasing its potential for both real-time emulation and biohybrid experiments. It presented how the BiœmuS is capable of emulating biophysically detailed models from single neurons to complex networks. It presented the different versions of the system developed tackling the intermediate and alternative versions as well as future improved versions with BiœmuM. Additionally, this section promotes the versatility of BiœmuS to integrate biohybrid closed-loop experiments and shows promising preliminary results for its capacity to embody the role of a main component in an electroceutical device.

Conclusion

The central theme addressed in this thesis was the design of a real-time biohybrid system based on a Spiking Neural network for biomedical applications targeting neurological disorders studies through real-time emulation and hybridization. The real-time emulation of biophysically detailed models comes as a viable alternative to existing emulation software thanks to an accessible software acquaintance, affordable price, flexibility and fast emulation. The hybridization served a significant purpose by contributing to the development of neuroprostheses, primarily through its ability to integrate biohybrid closed-loop systems based on the electroceutic approach, thus enhancing the interaction between artificial and biological neural networks. This manuscript was structured into four chapters, sequentially presenting the biological and technological background, the methods employed to develop the system and concluding with the integration of the system into applications and biohybrid experiments.

The starting point corresponded to a basic introduction of the functions, roles and characteristics of the different cells involved in the nervous system. The exploration of essential elements such as neurons and synapses, along with an understanding of their properties, has contributed to provide a good understanding of neurological disorders, which were subsequently introduced. Starting from their growing societal impact, the effect of neurological disorders at human scale and the cellular level as well as their insufficient management were detailed, emphasizing the need for efficient treatments. Alternative treatments introduced the electroceutical approach and neuroprostheses, emphasizing the significance of models, notably in the form of biomimetic neural networks. The concept of modeling raises questions about coherence, that translate in numeric systems in a trade-off between detail level and implementation cost. With regard to the level of detail, it allowed for the introduction of various models, each characterized by its inherent complexity and biological meaningfulness. This notably promoted the multicompartmental model as the most suitable for biomimetic purposes, this latter allowing an accurate modeling of both the electrophysiological and the geometrical properties of the neuron.

Stepping into the realm of numeric systems, as enforced by the concept of artificial modeling, an introduction to the numerical platform hosting the system was outlined. Following an overview of the technological context that supported the choice of a platform integrating a mixed architecture of CPU and FPGA, the main platform selected as well as its development methods were presented. Supplementary platforms used either in intermediate or alternative versions were also introduced. In line with the development of embedded systems, especially in the case of a flexible, accessible, and versatile system, its interconnection justified the need for an introduction to communication protocols and interfaces that play a crucial role. From the overview of the selected platform, two main parts stood out promoting the strength and weaknesses embodied by the two parts of the platform that can be roughly summed up as software for the PS and hardware for the PL.

As the groundwork for both the biological and technological background was established, the methods leading to the development of the different versions of the system developed were presented. The methods were dividing into two main categories being hardware and software that tackled the development of the computation core, monitoring and configuration of the system. The architecture of the computation core was described by addressing the implementation of the ion channels using pre-computed tables stored in memory. Then, it mostly focused on the architecture of the single compartment neurons computation that implements a mixed fixed-point and floating-point data coding to obtain a satisfying trade-off between resource utilization, performances and accuracy. An improved version of the computation core allowing the modeling of multicompartmental neurons was also presented along with optimization leads. Afterwards, the critical part of the computation core embodied by synapses was detailed, thus

showing the approach adopted to obtain satisfying performances. Next, the software presented responsible for the monitoring and configuration of the system was introduced, supporting the choices made to obtain a compromise between flexibility, accessibility and performances. The structure of the performant monitoring, control and configuration performed by the C++ application as well as the operating system running on the platform is notably introduced. Thereafter, the user-specific interfaces and scripts developed in Python were presented, promoting the accessibility of the system. Finally, the performances obtained were stated and compared to the others platforms, thus showing that the system developed shows promising performances. Most notably, the resource utilization is shown significantly low when projected on larger boards and the latency on a consistent timescale for the online monitoring of artificial neural networks.

With all the keys in hand, the full potential of the system was showcased through applications and biohybrid experiments, making a valuable contribution to the scientific community. Starting from the real-time emulation of independent biophysically detailed neuron to complex neural networks, the system demonstrated its capacity and potential to emulate biomimetic models. Especially, the preliminary version of the multicompartmental modeling in real-time on FPGA constitutes a promising contribution thanks to the efficient trade-off demonstrated between performances and resource utilization. Above all, the system best stood out when integrated in closed-loop and open-loop biohybrid experiments setup. It particularly demonstrated a great utility in the post-stroke rehabilitation through electroceutical therapy where it drove an adaptive stimulation that was capable of impacting the network activity. On a different level, it allowed the design of preliminary experiments demonstrating the feasibility of artificial synapses through optogenetic, thus constituting a great contribution to the development of neuroprostheses. Lastly, but not insignificantly, it showcases a highly promising interconnection with recent biophysical interfaces to obtain a biohybrid closed-loop targeting large artificial and biological neural networks, leading the way to greater investigation such as organoid intelligence.

Although the results presented in this thesis are promising, they represent preliminary work based mostly on the feasibility of the solutions and bridging the "hardware" gap between hybridization. Indeed, unraveling the mechanisms of learning in biological neural networks remains a significant challenge, requiring extensive investigation into communication with these networks, especially in the context of training in view of biological computing. Nonetheless, the findings discussed in this manuscript are essential to the establishment of a bidirectional communication with biological neural network, a crucial step in the realization of neuroprostheses.

From a boarder perspective, the thesis contributed to the exploration of the applications of FPGA and more specifically to mixed architecture integrating FPGA. Certainly, similarly to GPU, FPGA are significantly benefiting from the technological advances, further enhanced by recent architectures such as the adaptive SoCs proposed by AMD Xilinx, which integrate substantially higher computational power. Hence, as showcased by this work, FPGA-based architectures constitute valuable asset for real-time embedded systems.

In line with the discussion of recent architectures, leads of improvements would rely on the migration of the system on a more recent architecture that allow efficient floating-point handling in hardware. More importantly, great improvements would include the optimization on the software side such as the utilization of the real-time processors concurrently to the main cores running the operating system. Moreover, the implementation of a customized operating system enabling preemptive capabilities would represent a significant enhancement for the sys-

tem. A key area for improvement lies in optimizing synapses with a more efficient memory architecture like HBM, which can alleviate the current system bottleneck.

In conclusion, this thesis represents a promising preliminary work aligned with our team's primary objective, which is to advance towards the development of neuroprostheses as an effective treatment for neurological disorders. Ultimately, it is my aspiration that this work will become a valuable contribution to the scientific community, encouraging further enhancements and advancements.

Publications

Scientific Journals

Beaubois, R., Cheslet, J., Duenki, T., Khoyratee, F., Branchereau, P., Ikeuchi, Y., and Lévi, T. (2023). Biœmus: A new tool for neurological disorders studies through real-time emulation and hybridization using biomimetic spiking neural network. *(Under Review) Nature Communications*

Osaki, T., Beaubois, R., Lévi, T., Hirano, Y., and Ikeuchi, Y. (2023). Advanced complexity and plasticity of neural activity in reciprocally connected human cerebral organoids. *(Under Review) Nature Communications*

Chiappalone, M., Cota, V. R., Carè, M., Florio, M. D., Beaubois, R., Buccelli, S., Barban, F., Brofiga, M., Averna, A., Bonacini, F., Guggenmos, D. J., Bornat, Y., Massobrio, P., Bonifazi, P., and Levi, T. (2022). Neuromorphic-based neuroprostheses for brain rewiring: State-of-the-art and perspectives in neuroengineering. *Brain Sciences*, 12

International conferences

Di Florio, M., Carè, M., Beaubois, R., Barban, F., Levi, T., and Chiappalone, M. (2023). Design of an experimental setup for delivering intracortical microstimulation in vivo via spiking neural network. In *2023 45th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. IEEE

Beaubois, R., Khoyratee, F., Branchereau, P., Ikeuchi, Y., and Levi, T. (2022). From real-time single to multicompartmental Hodgkin-Huxley neurons on FPGA for bio-hybrid systems. In *2023 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 1602-1606

Beaubois, R., Hutsebaut, M., Molla, D., Ikeuchi, Y., Guilhem, L., and Timothée, L. (2022). Creation of neural melodies in the frame of an international student project. In *2022 31st Annual Conference of the European Association for Education in Electrical and Information Engineering (EAEEIE)*, pages 1-5. IEEE

National conferences

Levi, T., Beaubois, R., Hutsebaut, M., Molla, D., Ikeuchi, Y., and Guilhem, L. (2023). Création d'une musique neuronale. In *CNRIUT 2023*, France

Beaubois, R., Branchereau, P., Ikeuchi, Y., and Levi, T. (2023). Conception de neurones artificiels pour la réalisation d'expériences bio-hybrides : Étude des maladies neurodégénératives. In *Journée Francophone de la Recherche JFR 2023*, Japan

Beaubois, R., Branchereau, P., Ikeuchi, Y., and Levi, T. (2021). Conception de neurones artificiels pour la réalisation d'expériences bio-hybrides : Étude des maladies neurodégénératives. In *CNRIUT 2021*, France

Khoyratee, F., Beaubois, R., Larrieu, G., Ikeuchi, Y., and Lévi, T. (2020). Neurostimulation: Des expériences hybrides pour le futur du biomédical. In *Journée Francophone de la Recherche JFR 2020*, Japan

Bibliography

References

[And, 2018] (2018). Therapeutic decisions in ALS patients: cross-cultural differences and clinical implications. *Journal of Neurology*, 265(7):1600–1606.

[Abbott, 1999] Abbott, L. F. (1999). Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain research bulletin*, 50(5-6):303–304.

[Abbott and Nelson, 2000] Abbott, L. F. and Nelson, S. B. (2000). Synaptic plasticity: taming the beast. *Nature neuroscience*, 3(11):1178–1183.

[Abeles et al., 1993] Abeles, M., Bergman, H., Margalit, E., and Vaadia, E. (1993). Spatiotemporal firing patterns in the frontal cortex of behaving monkeys. *Journal of neurophysiology*, 70(4):1629–1638.

[Abeles and Gat, 2001] Abeles, M. and Gat, I. (2001). Detecting precise firing sequences in experimental data. *Journal of neuroscience methods*, 107(1-2):141–154.

[Adrian, 1926] Adrian, E. D. (1926). The impulses produced by sensory nerve endings: Part i. *The Journal of physiology*, 61(1):49.

[Akar et al., 2023] Akar, N. A., Biddiscombe, J., Cumming, B., Kabic, M., Karakasis, V., Klijn, W., Küsters, A., Peyser, A., Yates, S., Hater, T., Huisman, B., Hagen, E., Schepper, R. D., Linssen, C., Stoppels, H., Schmitt, S., Huber, F., Engelen, M., Bösch, F., Luboeinski, J., Frasch, S., Drescher, L., and Landsmeer, L. (2023). Arbor library v0.9.0.

[Akbarzadeh-Sherbaf et al., 2018] Akbarzadeh-Sherbaf, K., Abdoli, B., Safari, S., and Vahabie, A.-H. (2018). A scalable fpga architecture for randomly connected networks of hodgkin-huxley neurons. *Frontiers in Neuroscience*, 12:698.

[Amari, 1972] Amari, S.-I. (1972). Characteristics of random nets of analog neuron-like elements. *IEEE Transactions on systems, man, and cybernetics*, (5):643–657.

[Ambroise et al., 2013] Ambroise, M., Levi, T., Joucla, S., Yvert, B., and Saïghi, S. (2013). Real-time biomimetic central pattern generators in an fpga for hybrid experiments. *Frontiers in neuroscience*, 7:215.

[Amunts et al., 2016] Amunts, K., Ebell, C., Muller, J., Telefont, M., Knoll, A., and Lippert, T. (2016). The human brain project: Creating a european research infrastructure to decode the human brain. *Neuron*, 92(3):574–581.

[Andersen et al., 1992] Andersen, B. B., Korbo, L., and Pakkenberg, B. (1992). A quantitative study of the human cerebellum with unbiased stereological techniques. *Journal of Comparative Neurology*, 326(4):549–560.

[Antal et al., 2014] Antal, A., Bikson, M., Datta, A., Lafon, B., Dechent, P., Parra, L. C., and Paulus, W. (2014). Imaging artifacts induced by electrical stimulation during conventional fmri of the brain. *Neuroimage*, 85:1040–1047.

[Arora, 2012] Arora, M. (2012). The architecture and evolution of cpu-gpu systems for general purpose computing. In *Computer Science*.

[Averna et al., 2020] Averna, A., Pasquale, V., Murphy, M. D., Rogantin, M. P., Van Acker, G. M., Nudo, R. J., Chiappalone, M., and Guggenmos, D. J. (2020). Differential effects of open-and closed-loop intracortical microstimulation on firing patterns of neurons in distant cortical areas. *Cerebral Cortex*, 30(5):2879–2896.

[Ballard et al., 2006] Ballard, C. G., Waite, J., and Birks, J. (2006). Atypical antipsychotics for aggression and psychosis in alzheimer's disease. *Cochrane Database of Systematic Reviews*, (1).

[Ballini et al., 2014] Ballini, M., Müller, J., Livi, P., Chen, Y., Frey, U., Stettler, A., Shadmani, A., Viswam, V., Jones, I. L., Jäckel, D., et al. (2014). A 1024-channel cmos microelectrode array with 26,400 electrodes for recording and stimulation of electrogenic cells in vitro. *IEEE journal of solid-state circuits*, 49(11):2705–2719.

[Bateup et al., 2013] Bateup, H. S., Johnson, C. A., Denefrio, C. L., Saulnier, J. L., Kornacker, K., and Sabatini, B. L. (2013). Excitatory/inhibitory synaptic imbalance leads to hippocampal hyperexcitability in mouse models of tuberous sclerosis. *Neuron*, 78(3):510–522.

[Beaubois et al., 2022] Beaubois, R., Khoyratee, F., Branchereau, P., Ikeuchi, Y., and Levi, T. (2022). From real-time single to multicompartmental hodgkin-huxley neurons on fpga for bio-hybrid systems. In *2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 1602–1606.

[Ben-Ari et al., 1981] Ben-Ari, Y., Tremblay, E., Riche, D., Ghilini, G., and Naquet, R. (1981). Electrographic, clinical and pathological alterations following systemic administration of kainic acid, bicuculline or pentetrazole: metabolic mapping using the deoxyglucose method with special reference to the pathology of epilepsy. *Neuroscience*, 6(7):1361–1391.

[Bennett et al., 2019] Bennett, S., Laver, K., Voigt-Radloff, S., Letts, L., Clemson, L., Graff, M., Wiseman, J., and Gitlin, L. (2019). Occupational therapy for people with dementia and their family carers provided at home: a systematic review and meta-analysis. *BMJ open*, 9(11):e026308.

[Binczak et al., 2006] Binczak, S., Jacquir, S., Bilbault, J.-M., Kazantsev, V. B., and Nekorkin, V. I. (2006). Experimental study of electrical fitzhugh–nagumo neurons with modified excitability. *Neural Networks*, 19(5):684–693.

[Blanchard et al., 2019] Blanchard, D., Aihara, K., and Levi, T. (2019). Snake robot controlled by biomimetic cpgs. In *International Conference on Artificial Life and Robotics (ICAROB 2019)*. ALife Robotics Corporation Ltd.

[Bologna et al., 2010] Bologna, L. L., Pasquale, V., Garofalo, M., Gandolfo, M., Baljon, P. L., Maccione, A., Martinoia, S., and Chiappalone, M. (2010). Investigating neuronal activity by spycode multi-channel data analyzer. *Neural Networks*, 23(6):685–697.

[Bonifazi et al., 2013] Bonifazi, P., Difato, F., Massobrio, P., Breschi, G. L., Pasquale, V., Levi, T., Goldin, M., Bornat, Y., Tedesco, M., Bisio, M., et al. (2013). In vitro large-scale experimental and theoretical studies for the realization of bi-directional brain-prostheses. *Frontiers in neural circuits*, 7:40.

[Bono and Clopath, 2017] Bono, J. and Clopath, C. (2017). Modeling somatic and dendritic spike mediated plasticity at the single neuron and network level. *Nature communications*, 8(1):706.

[Bouton et al., 2016] Bouton, C. E., Shaikhouni, A., Annetta, N. V., Bockbrader, M. A., Friedenberg, D. A., Nielson, D. M., Sharma, G., Sederberg, P. B., Glenn, B. C., Mysiw, W. J., et al. (2016). Restoring cortical control of functional movement in a human with quadriplegia. *Nature*, 533(7602):247–250.

[Boutros and Betz, 2021] Boutros, A. and Betz, V. (2021). Fpga architecture: Principles and progression. *IEEE Circuits and Systems Magazine*, 21(2):4–29.

[Branchereau et al., 2016] Branchereau, P., Cattaert, D., Delpy, A., Allain, A.-E., Martin, E., and Meyrand, P. (2016). Depolarizing gaba/glycine synaptic events switch from excitation to inhibition during frequency increases. *Scientific Reports*, 6:21753.

[Branchereau et al., 2019] Branchereau, P., Martin, E., Allain, A.-E., Cazenave, W., Supiot, L., Hodeib, F., Laupénie, A., Dalvi, U., Zhu, H., and Cattaert, D. (2019). Relaxation of synaptic inhibitory events as a compensatory mechanism in fetal sod spinal motor networks. *Elife*, 8:e51402.

[Breslin and O'Driscoll, 2013] Breslin, S. and O'Driscoll, L. (2013). Three-dimensional cell culture: the missing link in drug discovery. *Drug discovery today*, 18(5-6):240–249.

[Brette, 2015a] Brette, R. (2015a). Philosophy of the spike: rate-based vs. spike-based theories of the brain. *Frontiers in systems neuroscience*, 9:151.

[Brette, 2015b] Brette, R. (2015b). What is the most realistic single-compartment model of spike initiation? *PLoS computational biology*, 11(4):e1004114.

[Brewer et al., 2008] Brewer, G. J., Boehler, M. D., Jones, T. T., and Wheeler, B. C. (2008). Nbactiv4 medium improvement to neurobasal/b27 increases neuron synapse densities and network spike rates on multielectrode arrays. *Journal of neuroscience methods*, 170(2):181–187.

[Broccard et al., 2017] Broccard, F. D., Joshi, S., Wang, J., and Cauwenberghs, G. (2017). Neuromorphic neural interfaces: from neurophysiological inspiration to biohybrid coupling with nervous systems. *Journal of neural engineering*, 14(4):041002.

[Brown, 1914] Brown, T. G. (1914). On the nature of the fundamental activity of the nervous centres; together with an analysis of the conditioning of rhythmic activity in progression, and a theory of the evolution of function in the nervous system. *The Journal of physiology*, 48(1):18.

[Buccelli et al., 2019] Buccelli, S., Bornat, Y., Colombi, I., Ambroise, M., Martines, L., Pasquale, V., Bisio, M., Tessadori, J., Nowak, P., Grassia, F., et al. (2019). A neuromorphic prosthesis to restore communication in neuronal networks. *IScience*, 19:402–414.

[Carè et al., 2022] Carè, M., Averna, A., Barban, F., Semprini, M., De Michieli, L., Nudo, R. J., Guggenmos, D. J., and Chiappalone, M. (2022). The impact of closed-loop intracortical stimulation on neural activity in brain-injured, anesthetized animals. *Bioelectronic Medicine*, 8(1):4.

[Carnevale and Hines, 2006] Carnevale, N. T. and Hines, M. L. (2006). *The NEURON book*. Cambridge University Press.

[Cassidy et al., 2011] Cassidy, A., Andreou, A. G., and Georgiou, J. (2011). Design of a one million neuron single fpga neuromorphic system for real-time multimodal scene analysis. In *2011 45Th annual conference on information sciences and systems*, pages 1–6. IEEE.

[Checkoway et al., 2011] Checkoway, H., Lundin, J. I., and Kelada, S. N. (2011). Neurodegenerative diseases. *IARC scientific publications*, (163):407–419.

[Chiappalone et al., 2022] Chiappalone, M., Cota, V. R., Carè, M., Florio, M. D., Beaubois, R., Buccelli, S., Barban, F., Brofiga, M., Averna, A., Bonacini, F., Guggenmos, D. J., Bornat, Y., Massobrio, P., Bonifazi, P., and Levi, T. (2022). Neuromorphic-based neuroprostheses for brain rewiring: State-of-the-art and perspectives in neuroengineering. *Brain Sciences*, 12.

[Chiappalone and Semprini, 2022] Chiappalone, M. and Semprini, M. (2022). Using robots to advance clinical translation in neurorehabilitation. *Science Robotics*, 7(64):eabo1966.

[Chin and Vora, 2014] Chin, J. H. and Vora, N. (2014). The global burden of neurologic diseases. *Neurology*, 83(4):349–351.

[Chklovskii et al., 2002] Chklovskii, D. B., Schikorski, T., and Stevens, C. F. (2002). Wiring optimization in cortical circuits. *Neuron*, 34(3):341–347.

[Choi et al., 2014] Choi, S. H., Kim, Y. H., Hebisch, M., Sliwinski, C., Lee, S., D'Avanzo, C., Chen, H., Hooli, B., Asselin, C., Muffat, J., et al. (2014). A three-dimensional human neural cell culture model of alzheimer's disease. *Nature*, 515(7526):274–278.

[Christensen et al., 2022] Christensen, D. V., Dittmann, R., Linares-Barranco, B., Sebastian, A., Le Gallo, M., Redaelli, A., Slesazeck, S., Mikolajick, T., Spiga, S., Menzel, S., et al. (2022). 2022 roadmap on neuromorphic computing and engineering. *Neuromorphic Computing and Engineering*, 2(2):022501.

[Cohen et al., 1992] Cohen, A. H., Ermentrout, G. B., Kiemel, T., Kopell, N., Sigvardt, K. A., and Williams, T. L. (1992). Modelling of intersegmental coordination in the lamprey central pattern generator for locomotion. *Trends in neurosciences*, 15(11):434–438.

[Cooper, 2011] Cooper, D. (2011). *Introduction to Neuroscience I.* Donald C. Cooper Ph.D.

[Corradi and Indiveri, 2015] Corradi, F. and Indiveri, G. (2015). A neuromorphic event-based neural recording system for smart brain-machine-interfaces. *IEEE transactions on biomedical circuits and systems*, 9(5):699–709.

[Cota-Coronado et al., 2019] Cota-Coronado, A., Díaz-Martínez, N. F., Padilla-Camberos, E., and Díaz-Martínez, N. E. (2019). Editing the central nervous system through crispr/cas9 systems. *Frontiers in molecular neuroscience*, 12:110.

[Cottone et al., 2018] Cottone, C., Cancelli, A., Pasqualetti, P., Porcaro, C., Salustri, C., and Tecchio, F. (2018). A new, high-efficacy, noninvasive transcranial electric stimulation tuned to local neurodynamics. *Journal of Neuroscience*, 38(3):586–594.

[Courant et al., 1967] Courant, R., Friedrichs, K., and Lewy, H. (1967). On the partial difference equations of mathematical physics. *IBM journal of Research and Development*, 11(2):215–234.

[Crockett et al., 2014] Crockett, L. H., Elliot, R. A., Enderwitz, M. A., and Stewart, R. W. (2014). *The Zynq book: embedded processing with the ARM Cortex-A9 on the Xilinx Zynq-7000 all programmable SoC.* Strathclyde Academic Media.

[Dally et al., 2021] Dally, W. J., Keckler, S. W., and Kirk, D. B. (2021). Evolution of the graphics processing unit (gpu). *IEEE Micro*, 41(6):42–51.

[Davidson and Furber, 2021] Davidson, S. and Furber, S. B. (2021). Comparison of artificial and spiking neural networks on digital hardware. *Frontiers in Neuroscience*, 15:651141.

[Davies et al., 2018] Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., Dimou, G., Joshi, P., Imam, N., Jain, S., et al. (2018). Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99.

[Debanne et al., 2011] Debanne, D., Campanac, E., Bialowas, A., Carlier, E., and Alcaraz, G. (2011). Axon physiology. *Physiological reviews*, 91(2):555–602.

[Destexhe et al., 1998] Destexhe, A., Mainen, Z. F., and Sejnowski, T. J. (1998). Kinetic models of synaptic transmission: From ions to networks. *Methods in Neural Modeling: from Ions to Networks*, pages 1–25.

[Destexhe et al., 2001] Destexhe, A., Rudolph, M., Fellous, J. M., and Sejnowski, T. J. (2001). Fluctuating synaptic conductances recreate in vivo-like activity in neocortical neurons. *Neuroscience*, 107:13–24.

[Deuschl et al., 2006] Deuschl, G., Schade-Brittinger, C., Krack, P., Volkmann, J., Schäfer, H., Bötzel, K., Daniels, C., Deutschländer, A., Dillmann, U., Eisner, W., et al. (2006). A Randomized Trial of Deep-Brain Stimulation for Parkinson's Disease. *New England Journal of Medicine*, 355(9):896–908.

[Di Florio et al., 2023] Di Florio, M., Carè, M., Beaubois, R., Barban, F., Levi, T., and Chiappalone, M. (2023). Design of an experimental setup for delivering intracortical microstimulation in vivo via spiking neural network. In *2023 45th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. IEEE.

[Ding et al., 2022] Ding, C., Wu, Y., Chen, X., Chen, Y., Wu, Z., Lin, Z., Kang, D., Fang, W., and Chen, F. (2022). Global, regional, and national burden and attributable risk factors of neurological disorders: The global burden of disease study 1990–2019. *Frontiers in Public Health*, 10:952161.

[Ding et al., 2021] Ding, Y., Fu, W., and Cao, L. (2021). Implementation of multi-compartment neuron model with synaptic plasticity using fpga. In *2021 IEEE 21st International Conference on Communication Technology (ICCT)*, pages 1008–1012. IEEE.

[Donati et al., 2019] Donati, E., Payvand, M., Risi, N., Krause, R., and Indiveri, G. (2019). Discrimination of emg signals using a neuromorphic implementation of a spiking neural network. *IEEE transactions on biomedical circuits and systems*, 13(5):795–803.

[Duval et al., 2017] Duval, K., Grover, H., Han, L.-H., Mou, Y., Pegoraro, A. F., Fredberg, J., and Chen, Z. (2017). Modeling physiological events in 2d vs. 3d cell culture. *Physiology*, 32(4):266–277.

[Elder et al., 2010] Elder, G. A., Gama Sosa, M. A., and De Gasperi, R. (2010). Transgenic mouse models of alzheimer's disease. *Mount Sinai Journal of Medicine: A Journal of Translational and Personalized Medicine: A Journal of Translational and Personalized Medicine*, 77(1):69–81.

[Espuny-Camacho et al., 2017] Espuny-Camacho, I., Arranz, A. M., Fiers, M., Snellinx, A., Ando, K., Munck, S., Bonnefont, J., Lambot, L., Corthout, N., Omodho, L., et al. (2017). Hallmarks of alzheimer's disease in stem-cell-derived human neurons transplanted into mouse brain. *Neuron*, 93(5):1066–1081.

[Ess, 2010] Ess, K. C. (2010). Tuberous sclerosis complex: a brave new world? *Current opinion in neurology*, 23(2):189.

[Famm et al., 2013] Famm, K., Litt, B., Tracey, K. J., Boyden, E. S., and Slaoui, M. (2013). A jump-start for electroceuticals. *Nature*, 496:159–161.

[Farina et al., 2021] Farina, D., Vujaklija, I., Brånemark, R., Bull, A. M., Dietl, H., Graimann, B., Hargrove, L. J., Hoffmann, K.-P., Huang, H., Ingvarsson, T., et al. (2021). Toward higher-performance bionic limbs for wider clinical use. *Nature biomedical engineering*, pages 1–13.

[FitzHugh, 1955] FitzHugh, R. (1955). Mathematical models of threshold phenomena in the nerve membrane. *The bulletin of mathematical biophysics*, 17:257–278.

[FitzHugh, 1961] FitzHugh, R. (1961). Impulses and physiological states in theoretical models of nerve membrane. *Biophysical journal*, 1(6):445–466.

[Fogarty et al., 2016a] Fogarty, M. J., Klenowski, P. M., Lee, J. D., Drieberg-Thompson, J. R., Bartlett, S. E., Ngo, S. T., Hilliard, M. A., Bellingham, M. C., and Noakes, P. G. (2016a). Cortical synaptic and dendritic spine abnormalities in a presymptomatic tdp-43 model of amyotrophic lateral sclerosis. *Scientific reports*, 6(1):37968.

[Fogarty et al., 2016b] Fogarty, M. J., Mu, E. W. H., Noakes, P. G., Lavidis, N. A., and Bellingham, M. C. (2016b). Marked changes in dendritic structure and spine density precede significant neuronal death in vulnerable cortical pyramidal neuron populations in the sod1(g93a) mouse model of amyotrophic lateral sclerosis. *Acta neuropathologica communications*, 4:77.

[Forrest et al., 2018] Forrest, M. P., Parnell, E., and Penzes, P. (2018). Dendritic structural plasticity and neuropsychiatric disease. *Nature Reviews Neuroscience*, 19(4):215–234.

[Fourcaud-Trocmé et al., 2003] Fourcaud-Trocmé, N., Hansel, D., Van Vreeswijk, C., and Brunel, N. (2003). How spike generation mechanisms determine the neuronal response to fluctuating inputs. *Journal of neuroscience*, 23(37):11628–11640.

[Frank, 2014] Frank, S. (2014). Treatment of huntington's disease. *Neurotherapeutics*, 11:153–160.

[French et al., 2016] French, B., Thomas, L. H., Coupe, J., McMahon, N. E., Connell, L., Harrison, J., Sutton, C. J., Tishkovskaya, S., and Watkins, C. L. (2016). Repetitive task training for improving functional ability after stroke. *Cochrane database of systematic reviews*, (11).

[Froemke et al., 2005] Froemke, R. C., Poo, M.-m., and Dan, Y. (2005). Spike-timing-dependent synaptic plasticity depends on dendritic location. *Nature*, 434(7030):221–225.

[Furue et al., 2007] Furue, H., Katafuchi, T., and Yoshimura, M. (2007). *In Vivo Patch-Clamp Technique*, pages 229–251. Humana Press, Totowa, NJ.

[Garcia et al., 1995] Garcia, J. H., Wagner, S., Liu, K.-F., and Hu, X.-j. (1995). Neurological deficit and extent of neuronal necrosis attributable to middle cerebral artery occlusion in rats: statistical validation. *Stroke*, 26(4):627–635.

[Gasparini et al., 2004] Gasparini, S., Migliore, M., and Magee, J. C. (2004). On the initiation and propagation of dendritic spikes in ca1 pyramidal neurons. *Journal of Neuroscience*, 24(49):11046–11056.

[Gauthier et al., 2016] Gauthier, S., Feldman, H. H., Schneider, L. S., Wilcock, G. K., Frisoni, G. B., Hardlund, J. H., Moebius, H. J., Bentham, P., Kook, K. A., Wischik, D. J., et al. (2016). Efficacy and safety of tau-aggregation inhibitor therapy in patients with mild or

moderate alzheimer's disease: a randomised, controlled, double-blind, parallel-arm, phase 3 trial. *The Lancet*, 388(10062):2873–2884.

[George et al., 2020] George, R., Chiappalone, M., Giugliano, M., Levi, T., Vassanelli, S., Partzsch, J., and Mayr, C. (2020). Plasticity and adaptation in neuromorphic biohybrid systems. *Iscience*, 23(10).

[George et al., 2013] George, S., Hasler, J., Koziol, S., Nease, S., and Ramakrishnan, S. (2013). Low power dendritic computation for wordspotting. *Journal of Low Power Electronics and Applications*, 3(2):73–98.

[Gerstner and Brette, 2009] Gerstner, W. and Brette, R. (2009). Adaptive exponential integrate-and-fire model. *Scholarpedia*, 4(6):8427.

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

[Gewaltig and Diesmann, 2007] Gewaltig, M.-O. and Diesmann, M. (2007). Nest (neural simulation tool). *Scholarpedia*, 2(4):1430.

[Ghosh-Dastidar and Adeli, 2009] Ghosh-Dastidar, S. and Adeli, H. (2009). Spiking neural networks. *International journal of neural systems*, 19(04):295–308.

[Gobel and Helmchen, 2007] Gobel, W. and Helmchen, F. (2007). In vivo calcium imaging of neural network function. *Physiology*, 22(6):358–365.

[Grassia et al., 2016] Grassia, F., Kohno, T., and Levi, T. (2016). Digital hardware implementation of a stochastic two-dimensional neuron model. *Journal of Physiology Paris*, 110:409–416.

[Grienberger and Konnerth, 2012] Grienberger, C. and Konnerth, A. (2012). Imaging calcium in neurons. *Neuron*, 73(5):862–885.

[Guggenmos et al., 2013] Guggenmos, D. J., Azin, M., Barbay, S., Mahnken, J. D., Dunham, C., Mohseni, P., and Nudo, R. J. (2013). Restoration of function after brain damage using a neural prosthesis. *Proceedings of the National Academy of Sciences*, 110(52):21177–21182.

[Hasler et al., 2007] Hasler, P., Kozoil, S., Farquhar, E., and Basu, A. (2007). Transistor channel dendrites implementing hmm classifiers. In *2007 IEEE International Symposium on Circuits and Systems*, pages 3359–3362. IEEE.

[Hebb, 2005] Hebb, D. O. (2005). *The organization of behavior: A neuropsychological theory*. Psychology press.

[Herculano-Houzel, 2009] Herculano-Houzel, S. (2009). The human brain in numbers: a linearly scaled-up primate brain. *Frontiers in human neuroscience*, page 31.

[Hill et al., 2001] Hill, A. A., Lu, J., Masino, M., Olsen, O., and Calabrese, R. L. (2001). A model of a segmental oscillator in the leech heartbeat neuronal network. *Journal of computational neuroscience*, 10:281–302.

[Hines, 1984] Hines, M. (1984). Efficient computation of branched nerve equations. *International journal of bio-medical computing*, 15(1):69–76.

[Hines and Carnevale, 2001] Hines, M. L. and Carnevale, N. T. (2001). Neuron: a tool for neuroscientists. *The neuroscientist*, 7(2):123–135.

[Hochberg et al., 2012] Hochberg, L. R., Bacher, D., Jarosiewicz, B., Masse, N. Y., Simeral, J. D., Vogel, J., Haddadin, S., Liu, J., Cash, S. S., Van Der Smagt, P., et al. (2012). Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. *Nature*, 485(7398):372–375.

[Hodgkin and Huxley, 1990] Hodgkin, A. and Huxley, A. (1990). A quantitative description of membrane current and its application to conduction and excitation in nerve. *Bulletin of Mathematical Biology*, 52:25–71.

[Huh et al., 2011] Huh, D., Hamilton, G. A., and Ingber, D. E. (2011). From 3d cell culture to organs-on-chips. *Trends in cell biology*, 21(12):745–754.

[Ijspeert et al., 2007] Ijspeert, A. J., Crespi, A., Ryczko, D., and Cabelguen, J.-M. (2007). From swimming to walking with a salamander robot driven by a spinal cord model. *Science*, 315(5817):1416–1420.

[Indiveri and Fusi, 2007] Indiveri, G. and Fusi, S. (2007). Spike-based learning in vlsi networks of integrate-and-fire neurons. In *2007 IEEE international symposium on circuits and systems*, pages 3371–3374. IEEE.

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

[Izhikevich, 2004] Izhikevich, E. M. (2004). Which model to use for cortical spiking neurons? *IEEE transactions on neural networks*, 15(5):1063–1070.

[Izhikevich and Edelman, 2008] Izhikevich, E. M. and Edelman, G. M. (2008). Large-scale model of mammalian thalamocortical systems. *Proceedings of the national academy of sciences*, 105(9):3593–3598.

[Jackson et al., 2006] Jackson, A., Mavoori, J., and Fetz, E. E. (2006). Long-term motor cortex plasticity induced by an electronic neural implant. *Nature*, 444(7115):56–60.

[Jahnsen and Llinás, 1984] Jahnsen, H. and Llinás, R. (1984). Ionic basis for the electroresponsiveness and oscillatory properties of guinea-pig thalamic neurones in vitro. *The Journal of physiology*, 349(1):227–247.

[Jordan et al., 2018] Jordan, J., Ippen, T., Helias, M., Kitayama, I., Sato, M., Igarashi, J., Diesmann, M., and Kunkel, S. (2018). Extremely scalable spiking neuronal network simulation code: from laptops to exascale computers. *Frontiers in neuroinformatics*, 12:2.

[Julien and Kriz, 2006] Julien, J.-P. and Kriz, J. (2006). Transgenic mouse models of amyotrophic lateral sclerosis. *Biochimica et Biophysica Acta (BBA)-Molecular Basis of Disease*, 1762(11-12):1013–1024.

[Kagan et al., 2022] Kagan, B. J., Kitchen, A. C., Tran, N. T., Habibollahi, F., Khajehnejad, M., Parker, B. J., Bhat, A., Rollo, B., Razi, A., and Friston, K. J. (2022). In vitro neurons learn and exhibit sentience when embodied in a simulated game-world. *Neuron*, 110(23):3952–3969.

[Kandel et al., 2000] Kandel, E. R., Schwartz, J. H., Jessell, T. M., Siegelbaum, S., Hudspeth, A. J., Mack, S., et al. (2000). *Principles of neural science*, volume 4. McGraw-hill New York.

[Kawada et al., 2017] Kawada, J., Kaneda, S., Kirihara, T., Maroof, A., Levi, T., Eggan, K., Fujii, T., and Ikeuchi, Y. (2017). Generation of a motor nerve organoid with human stem cell-derived neurons. *Stem cell reports*, 9(5):1441–1449.

[Kernell and Zwaagstra, 1981] Kernell, D. and Zwaagstra, B. (1981). Input conductance, axonal conduction velocity and cell size among hindlimb motoneurones of the cat. *Brain Research*, 204(2):311–326.

[Kheradpisheh et al., 2018] Kheradpisheh, S. R., Ganjtabesh, M., Thorpe, S. J., and Masquelier, T. (2018). Stdp-based spiking deep convolutional neural networks for object recognition. *Neural Networks*, 99:56–67.

[Khoyratee et al., 2019] Khoyratee, F., Grassia, F., Saïghi, S., and Levi, T. (2019). Optimized real-time biomimetic neural network on fpga for bio-hybridization. *Frontiers in neuroscience*, 13:377.

[Kim et al., 2020] Kim, J., Koo, B.-K., and Knoblich, J. A. (2020). Human organoids: model systems for human biology and medicine. *Nature Reviews Molecular Cell Biology*, 21(10):571–584.

[Kirihara et al., 2019] Kirihara, T., Luo, Z., Chow, S. Y. A., Misawa, R., Kawada, J., Shibata, S., Khoyratee, F., Vollette, C. A., Volz, V., Levi, T., et al. (2019). A human induced pluripotent stem cell-derived tissue model of a cerebral tract connecting two cortical regions. *Iscience*, 14:301–311.

[Kobayashi et al., 2021] Kobayashi, T., Kuriyama, R., and Yamazaki, T. (2021). Testing an explicit method for multi-compartment neuron model simulation on a gpu. *Cognitive Computation*.

[Kohno et al., 2016] Kohno, T., Sekikawa, M., Li, J., Nanami, T., and Aihara, K. (2016). Qualitative-modeling-based silicon neurons and their networks. *Frontiers in neuroscience*, 10:273.

[Kolli et al., 2018] Kolli, N., Lu, M., Maiti, P., Rossignol, J., and Dunbar, G. L. (2018). Application of the gene editing tool, crispr-cas9, for treating neurodegenerative diseases. *Neurochemistry international*, 112:187–196.

[Koo et al., 2006] Koo, V., Hamilton, P., and Williamson, K. (2006). Non-invasive in vivo imaging in small animal research. *Analytical Cellular Pathology*, 28(4):127–139.

[Krogh, 2008] Krogh, A. (2008). What are artificial neural networks? *Nature biotechnology*, 26(2):195–197.

[Lai and Jan, 2006] Lai, H. C. and Jan, L. Y. (2006). The distribution and targeting of neuronal voltage-gated ion channels. *Nature Reviews Neuroscience*, 7(7):548–562.

[Larrieu and Han, 2013] Larrieu, G. and Han, X.-L. (2013). Vertical nanowire array-based field effect transistors for ultimate scaling. *Nanoscale*, 5(6):2437–2441.

[Le Masson et al., 2002] Le Masson, G., Renaud-Le Masson, S., Debay, D., and Bal, T. (2002). Feedback inhibition controls spike transfer in hybrid thalamic circuits. *Nature*, 417(6891):854–858.

[Levi et al., 2018] Levi, T., Bonifazi, P., Massobrio, P., and Chiappalone, M. (2018). editorial: closed-loop systems for next-generation neuroprostheses. *Frontiers in neuroscience*, 12.

[Liu and Douglas, 2004] Liu, S.-C. and Douglas, R. (2004). Temporal coding in a silicon network of integrate-and-fire neurons. *IEEE Transactions on neural networks*, 15(5):1305–1314.

[Lorenzo et al., 2000] Lorenzo, A., Yuan, M., Zhang, Z., Paganetti, P. A., Sturchler-Pierrat, C., Staufenbiel, M., Mautino, J., Vigo, F. S., Sommer, B., and Yankner, B. A. (2000). Amyloid β interacts with the amyloid precursor protein: a potential toxic mechanism in alzheimer's disease. *Nature neuroscience*, 3(5):460–464.

[Ma et al., 2021] Ma, C., Peng, Y., Li, H., and Chen, W. (2021). Organ-on-a-chip: a new paradigm for drug development. *Trends in pharmacological sciences*, 42(2):119–133.

[Maass et al., 2002] Maass, W., Natschläger, T., and Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560.

[Maccione et al., 2009] Maccione, A., Gandolfo, M., Massobrio, P., Novellino, A., Martinoia, S., and Chiappalone, M. (2009). A novel algorithm for precise identification of spikes in extracellularly recorded neuronal signals. *Journal of neuroscience methods*, 177(1):241–249.

[Marder and Bucher, 2001] Marder, E. and Bucher, D. (2001). Central pattern generators and the control of rhythmic movements. *Current biology*, 11(23):R986–R996.

[Martin et al., 2020] Martin, E., Cazenave, W., Allain, A.-E., Cattaert, D., and Branchereau, P. (2020). Implication of 5-ht in the dysregulation of chloride homeostasis in prenatal spinal motoneurons from the g93a mouse model of amyotrophic lateral sclerosis. *International journal of molecular sciences*, 21(3):1107.

[Martin et al., 2013] Martin, E., Cazenave, W., Cattaert, D., and Branchereau, P. (2013). Embryonic alteration of motoneuronal morphology induces hyperexcitability in the mouse model of amyotrophic lateral sclerosis. *Neurobiology of disease*, 54:116–126.

[Mayr et al., 2019] Mayr, C., Hoeppner, S., and Furber, S. (2019). Spinnaker 2: A 10 million core processor system for brain simulation and machine learning. *arXiv preprint arXiv:1911.02385*.

[McBain and Fisahn, 2001] McBain, C. J. and Fisahn, A. (2001). Interneurons unbound. *Nature Reviews Neuroscience*, 2(1):11–23.

[Medsker and Jain, 2001] Medsker, L. R. and Jain, L. (2001). Recurrent neural networks. *Design and Applications*, 5(64-67):2.

[Mel, 1999] Mel, B. W. (1999). Why have dendrites? a computational perspective. *Dendrites*, pages 271–89.

[Meldrum and Nilsson, 1976] Meldrum, B. S. and Nilsson, B. (1976). Cerebral blood flow and metabolic rate early and late in prolonged epileptic seizures induced in rats by bicuculline. *Brain: a journal of neurology*, 99(3):523–542.

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

[Miller et al., 2012] Miller, R. G., Mitchell, J. D., and Moore, D. H. (2012). Riluzole for amyotrophic lateral sclerosis (als)/motor neuron disease (mnd). *Cochrane database of systematic reviews*, (3).

[Morris and Lecar, 1981] Morris, C. and Lecar, H. (1981). Voltage oscillations in the barnacle giant muscle fiber. *Biophysical journal*, 35(1):193–213.

[Mosbacher et al., 2020] Mosbacher, Y., Khoyratee, F., Goldin, M., Kanner, S., Malakai, Y., Silva, M., Grassia, F., Simon, Y. B., Cortes, J., Barzilai, A., et al. (2020). Toward neuroprosthetic real-time communication from in silico to biological neuronal network via patterned optogenetic stimulation. *Scientific reports*, 10(1):7512.

[Müller et al., 2015] Müller, J., Ballini, M., Livi, P., Chen, Y., Radivojevic, M., Shadmani, A., Viswam, V., Jones, I. L., Fiscella, M., Diggelmann, R., et al. (2015). High-resolution cmos mea platform to study neurons at subcellular, cellular, and network levels. *Lab on a Chip*, 15(13):2767–2780.

[Mäki-M et al., 2018] Mäki-M, T., Halnes, G., Devor, A., Metzner, C., Dale, A. M., Andreassen, O. A., and Einevoll, G. T. (2018). A stepwise neuron model fitting procedure designed for recordings with high spatial resolution: Application to layer 5 pyramidal cells. *Journal of neuroscience methods*, 293:264–283.

[Nagel et al., 2003] Nagel, G., Szellas, T., Huhn, W., Kateriya, S., Adeishvili, N., Berthold, P., Ollig, D., Hegemann, P., and Bamberg, E. (2003). Channelrhodopsin-2, a directly light-gated cation-selective membrane channel. *Proceedings of the National Academy of Sciences*, 100(24):13940–13945.

[Nagumo et al., 1962] Nagumo, J., Arimoto, S., and Yoshizawa, S. (1962). An active pulse transmission line simulating nerve axon. *Proceedings of the IRE*, 50(10):2061–2070.

[Natarajan and Hasler, 2018] Natarajan, A. and Hasler, J. (2018). Hodgkin–huxley neuron and fpaa dynamics. *IEEE transactions on biomedical circuits and systems*, 12(4):918–926.

[Naundorf et al., 2006] Naundorf, B., Wolf, F., and Volgushev, M. (2006). Unique features of action potential initiation in cortical neurons. *Nature*, 440(7087):1060–1063.

[Nazari et al., 2015] Nazari, S., Faez, K., Amiri, M., and Karami, E. (2015). A digital implementation of neuron–astrocyte interaction for neuromorphic applications. *Neural Networks*, 66:79–90.

[Neher and Sakmann, 1992] Neher, E. and Sakmann, B. (1992). The patch clamp technique. *Scientific American*, 266(3):44–51.

[Nickolls and Dally, 2010] Nickolls, J. and Dally, W. J. (2010). The gpu computing era. *IEEE micro*, 30(2):56–69.

[Nicolelis and Lebedev, 2009] Nicolelis, M. A. and Lebedev, M. A. (2009). Principles of neural ensemble physiology underlying the operation of brain–machine interfaces. *Nature reviews neuroscience*, 10(7):530–540.

[Nishikawa et al., 2019] Nishikawa, S. M., Khoyratee, F., Kim, S. H., Ikeuchi, Y., Aihara, K., Fujii, T., and Levi, T. (2019). Biomimetic spike-timing based ionic microstimulation for neuron culture. In *ICAROB*.

[Nordhaus, 2007] Nordhaus, W. D. (2007). Two centuries of productivity growth in computing. *The Journal of Economic History*, 67(1):128–159.

[Norrby, 2006] Norrby, K. (2006). In vivo models of angiogenesis. *Journal of cellular and molecular medicine*, 10(3):588–612.

[Nudo et al., 1996] Nudo, R. J., Milliken, G. W., Jenkins, W. M., and Merzenich, M. M. (1996). Use-dependent alterations of movement representations in primary motor cortex of adult squirrel monkeys. *The Journal of neuroscience*, 16(2):785.

[Organization et al., 2020] Organization, W. H. et al. (2020). The top 10 causes of death; 24 may 2018.

[Osaki and Ikeuchi, 2021] Osaki, T. and Ikeuchi, Y. (2021). Advanced complexity and plasticity of neural activity in reciprocally connected human cerebral organoids. *BioRxiv*, pages 2021–02.

[O'Shea and Nash, 2015] O'Shea, K. and Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.

[Osorio, 2016] Osorio, R. R. (2016). Pipelined fpga implementation of numerical integration of the hodgkin-huxley model. In *2016 IEEE 27th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 202–206. IEEE.

[Pahan, 2019] Pahan, K. (2019). A broad application of crispr cas9 in infectious, inflammatory and neurodegenerative diseases. *Journal of Neuroimmune Pharmacology*, 14:534–536.

[Painkras et al., 2013] Painkras, E., Plana, L. A., Garside, J., Temple, S., Galluppi, F., Patterson, C., Lester, D. R., Brown, A. D., and Furber, S. B. (2013). Spinnaker: A 1-w 18-core system-on-chip for massively-parallel neural network simulation. *IEEE Journal of Solid-State Circuits*, 48(8):1943–1953.

[Pampaloni et al., 2007] Pampaloni, F., Reynaud, E. G., and Stelzer, E. H. (2007). The third dimension bridges the gap between cell culture and live tissue. *Nature reviews Molecular cell biology*, 8(10):839–845.

[Panuccio et al., 2018] Panuccio, G., Semprini, M., Natale, L., Buccelli, S., Colombi, I., and Chiappalone, M. (2018). Progress in neuroengineering for brain repair: New challenges and open issues. *Brain and neuroscience advances*, 2:2398212818776475.

[Paşca, 2019] Paşca, S. P. (2019). Assembling human brain organoids. *Science*, 363(6423):126–127.

[Patterson and Hennessy, 2013] Patterson, D. A. and Hennessy, J. L. (2013). *Computer Organization and Design, Fifth Edition: The Hardware/Software Interface*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edition.

[Pehle et al., 2022] Pehle, C., Billaudelle, S., Cramer, B., Kaiser, J., Schreiber, K., Stradmann, Y., Weis, J., Leibfried, A., Müller, E., and Schemmel, J. (2022). The brainscales-2 accelerated neuromorphic system with hybrid plasticity. *Frontiers in Neuroscience*, 16.

[Pelvig et al., 2008] Pelvig, D. P., Pakkenberg, H., Stark, A. K., and Pakkenberg, B. (2008). Neocortical glial cell numbers in human brains. *Neurobiology of aging*, 29(11):1754–1762.

[Peters et al., 2007] Peters, L. L., Robledo, R. F., Bult, C. J., Churchill, G. A., Paigen, B. J., and Svenson, K. L. (2007). The mouse as a model for human biology: a resource guide for complex trait analysis. *Nature Reviews Genetics*, 8(1):58–69.

[Pospischil et al., 2008] Pospischil, M., Toledo-Rodriguez, M., Monier, C., Piwkowska, Z., Bal, T., Frégnac, Y., Markram, H., and Destexhe, A. (2008). Minimal hodgkin-huxley type models for different classes of cortical and thalamic neurons. *Biological Cybernetics*, 99:427–441.

[PratapSingh and Kumar, 2014] PratapSingh, M. and Kumar, M. (2014). Evolution of processor architecture in mobile phones. *International Journal of Computer Applications*, 90.

[Qiao et al., 2015] Qiao, N., Mostafa, H., Corradi, F., Osswald, M., Stefanini, F., Sumislawska, D., and Indiveri, G. (2015). A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses. *Frontiers in neuroscience*, 9:141.

[Reardon, 2014] Reardon, S. (2014). Electroceuticals spark interest. *Nature*, 511:18.

[Regnell et al., 2012] Regnell, C. E., Hildrestrand, G. A., Sejersted, Y., Medin, T., Moldestad, O., Rolseth, V., Krokeide, S. Z., Suganthan, R., Luna, L., Bjørås, M., et al. (2012). Hippocampal adult neurogenesis is maintained by neil3-dependent repair of oxidative dna lesions in neural progenitor cells. *Cell reports*, 2(3):503–510.

[Renaud et al., 2007] Renaud, S., Tomas, J., Bornat, Y., Daouzli, A., and Saïghi, S. (2007). Neuromimetic ics with analog cores: an alternative for simulating spiking neural networks. In *2007 IEEE international symposium on circuits and systems*, pages 3355–3358. IEEE.

[Reuveni et al., 1993] Reuveni, I., Friedman, A., Amitai, Y., and Gutnick, M. J. (1993). Stepwise repolarization from ca2+ plateaus in neocortical pyramidal cells: evidence for nonhomogeneous distribution of hva ca2+ channels in dendrites. *Journal of Neuroscience*, 13(11):4609–4621.

[Rice et al., 2009] Rice, K. L., Bhuiyan, M. A., Taha, T. M., Vutsinas, C. N., and Smith, M. C. (2009). Fpga implementation of izhikevich spiking neural networks for character recognition. In *2009 International Conference on Reconfigurable Computing and FPGAs*, pages 451–456. IEEE.

[Rust et al., 2019] Rust, R., Grönnert, L., Gantner, C., Enzler, A., Mulders, G., Weber, R. Z., Siewert, A., Limasale, Y. D., Meinhardt, A., Maurer, M. A., et al. (2019). Nogo-a targeted therapy promotes vascular repair and functional recovery following stroke. *Proceedings of the National Academy of Sciences*, 116(28):14270–14279.

[Schemmel et al., 2007] Schemmel, J., Bruderle, D., Meier, K., and Ostendorf, B. (2007). Modeling synaptic plasticity within networks of highly accelerated i&f neurons. In *2007 IEEE international symposium on circuits and systems*, pages 3367–3370. IEEE.

[Seal, 2000] Seal, D. (2000). *ARM Architecture Reference Manual*. Addison-Wesley Longman Publishing Co., Inc., USA, 2nd edition.

[Semprini et al., 2018] Semprini, M., Laffranchi, M., Sanguineti, V., Avanzino, L., De Icco, R., De Michieli, L., and Chiappalone, M. (2018). Technological approaches for neurorehabilitation: from robotic devices to brain stimulation and beyond. *Frontiers in neurology*, 9:212.

[Sharifshazileh et al., 2021] Sharifshazileh, M., Burelo, K., Sarnthein, J., and Indiveri, G. (2021). An electronic neuromorphic system for real-time detection of high frequency oscillations (hfo) in intracranial eeg. *Nature communications*, 12(1):3095.

[Shu et al., 2011] Shu, L., Ueda, K., Chiu, I., and Cheong, H. (2011). Biologically inspired design. *CIRP annals*, 60(2):673–693.

[Smirnova et al., 2023] Smirnova, L., Caffo, B. S., Gracias, D. H., Huang, Q., Morales Pantoja, I. E., Tang, B., Zack, D. J., Berlinicke, C. A., Boyd, J. L., Harris, T. D., et al. (2023). Organoid intelligence (oi): the new frontier in biocomputing and intelligence-in-a-dish. *Frontiers in Science*, page 0.

[Smith et al., 2013] Smith, S. L., Smith, I. T., Branco, T., and Häusser, M. (2013). Dendritic spikes enhance stimulus selectivity in cortical neurons in vivo. *Nature*, 503(7474):115–120.

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

[Sorensen et al., 2004] Sorensen, M., DeWeerth, S., Cymbalyuk, G., and Calabrese, R. L. (2004). Using a hybrid neural system to reveal regulation of neuronal network activity by an intrinsic current. *Journal of Neuroscience*, 24(23):5427–5438.

[Spillane et al., 2016] Spillane, J., Kullmann, D., and Hanna, M. (2016). Genetic neurological channelopathies: molecular genetics and clinical phenotypes. *Journal of Neurology, Neurosurgery & Psychiatry*, 87(1):37–48.

[Spira and Hai, 2020] Spira, M. E. and Hai, A. (2020). Multi-electrode array technologies for neuroscience and cardiology. *Nano-Enabled Medical Applications*, pages 567–602.

[Stimberg et al., 2019] Stimberg, M., Brette, R., and Goodman, D. F. (2019). Brian 2, an intuitive and efficient neural simulator. *Elife*, 8:e47314.

[Stimberg et al., 2013] Stimberg, M., Goodman, D. F. M., Benichoux, V., and Brette, R. (2013). Brian 2 - the second coming: spiking neural network simulation in Python with code generation. *BMC Neuroscience*, 14(1):P38.

[Stradmann et al., 2022] Stradmann, Y., Billaudelle, S., Breitwieser, O., Ebert, F. L., Emmel, A., Husmann, D., Ilmberger, J., Müller, E., Spilger, P., Weis, J., et al. (2022). Demonstrating analog inference on the brainscales-2 mobile system. *IEEE Open Journal of Circuits and Systems*, 3:252–262.

[Svozil et al., 1997] Svozil, D., Kvasnicka, V., and Pospichal, J. (1997). Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 39(1):43–62.

[Tariot et al., 2004] Tariot, P. N., Farlow, M. R., Grossberg, G. T., Graham, S. M., McDonald, S., Gergel, I., Group, M. S., Group, M. S., et al. (2004). Memantine treatment in patients with moderate to severe alzheimer disease already receiving donepezil: a randomized controlled trial. *Jama*, 291(3):317–324.

[Tavanaei et al., 2019] Tavanaei, A., Ghodrati, M., Kheradpisheh, S. R., Masquelier, T., and Maida, A. (2019). Deep learning in spiking neural networks. *Neural networks*, 111:47–63.

[Tehovnik, 1996] Tehovnik, E. J. (1996). Electrical stimulation of neural tissue to evoke behavioral responses. *Journal of neuroscience methods*, 65(1):1–17.

[Thorpe et al., 1996] Thorpe, S., Fize, D., and Marlot, C. (1996). Speed of processing in the human visual system. *Nature*, 381(6582):520–522.

[Toledo-Rodriguez et al., 2004] Toledo-Rodriguez, M., Blumenfeld, B., Wu, C., Luo, J., Attali, B., Goodman, P., and Markram, H. (2004). Correlation maps allow neuronal electrical properties to be predicted from single-cell gene expression profiles in rat neocortex. *Cerebral cortex*, 14(12):1310–1327.

[Touboul, 2008] Touboul, J. (2008). Bifurcation analysis of a general class of nonlinear integrate-and-fire neurons. *SIAM Journal on Applied Mathematics*, 68(4):1045–1079.

[Tuckwell et al., 2002] Tuckwell, H. C., Wan, F. Y., and Rospars, J.-P. (2002). A spatial stochastic neuronal model with ornstein–uhlenbeck input current. *Biological cybernetics*, 86(2):137–145.

[Valentian et al., 2019] Valentian, A., Rummens, F., Vianello, E., Mesquida, T., de Boissac, C. L.-M., Bichler, O., and Reita, C. (2019). Fully integrated spiking neural network with analog neurons and rram synapses. In *2019 IEEE International Electron Devices Meeting (IEDM)*, pages 14–3. IEEE.

[Valero-Lara et al., 2018] Valero-Lara, P., Martínez-Pérez, I., Sirvent, R., Pena, A. J., Martorell, X., and Labarta, J. (2018). Simulating the behavior of the human brain on gpus. *Oil & Gas Science and Technology–Revue d'IFP Energies nouvelles*, 73:63.

[Van Albada et al., 2018] Van Albada, S. J., Rowley, A. G., Senk, J., Hopkins, M., Schmidt, M., Stokes, A. B., Lester, D. R., Diesmann, M., and Furber, S. B. (2018). Performance comparison of the digital neuromorphic hardware spinnaker and the neural network simulation software nest for a full-scale cortical microcircuit model. *Frontiers in neuroscience*, 12:291.

[Van Der Pol and Van Der Mark, 1928] Van Der Pol, B. and Van Der Mark, J. (1928). Lxxii. the heartbeat considered as a relaxation oscillation, and an electrical model of the heart. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 6(38):763–775.

[Vogelstein et al., 2004] Vogelstein, R. J., Mallik, U., and Cauwenberghs, G. (2004). Silicon spike-based synaptic array and address-event transceiver. In *2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No. 04CH37512)*, volume 5, pages V–V. IEEE.

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

[Xilinx, 2017] Xilinx, I. (2017). *Vivado Design Suite User Guide: Release Notes, Installation, and Licensing (UG973).*

[Xilinx, 2021] Xilinx, I. (2021). *UltraScale Architecture DSP Slice User Guide (UG579).*

[Xu et al., 2018] Xu, T., Xiao, N., Zhai, X., Chan, P. K., and Tin, C. (2018). Real-time cerebellar neuroprosthetic system based on a spiking neural network model of motor learning. *Journal of Neural Engineering*, 15(1):016021.

[Yaghini Bonabi et al., 2014] Yaghini Bonabi, S., Asgharian, H., Safari, S., and Nili Ahmadabadi, M. (2014). Fpga implementation of a biological neural network based on the hodgkin-huxley neuron model. *Frontiers in neuroscience*, 8:379.

[Yamada et al., 1989] Yamada, W., Koch, C., and Adams, P. (1989). Methods in neuronal modeling. *Cambridge, MA, USA.*

[Zhang et al., 2023] Zhang, Y., He, G., Ma, L., Liu, X., Hjorth, J. J., Kozlov, A., He, Y., Zhang, S., Kotaleski, J. H., Tian, Y., et al. (2023). A gpu-based computational framework that bridges neuron simulation and artificial intelligence. *Nature Communications*, 14(1):5798.