UNIVERSITY OF GENOVA

PHD PROGRAM IN BIOENGINEERING AND ROBOTICS

# Exploiting All-Programmable System on Chips for Closed-Loop Real-Time Neural Interfaces

by

**Giovanni Pietro Seu**

Thesis submitted for the degree of *Doctor of Philosophy* (31° cycle)

December 2018

| | |
|---|---|
| Paolo Meloni | Supervisor |
| Giorgio Cannata | Head of the PhD program |

***Thesis Jury:***

| | |
|---|---|
| Eduardo Ros, *Universidad de Granada* | External examiner |
| Eduardo de la Torre, *Universidad Politécnica de Madrid* | External examiner |

Dibris

Department of Informatics, Bioengineering, Robotics and Systems Engineering

*To my parents and my grandmothers who taught me to always do my best.*

*To Antonio Gramsci whose life must never be forgotten*
*and whose death must never be forgiven.*

# Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Giovanni Pietro Seu<br>
April 2019

# Acknowledgements

This research has received funding by the EU Commission's H2020 Programme under grant agreement N. 732105 (CERBERO project) and EU Commission's Seventh Framework Programme for Research (FP7) under grant agreement N. 248424 (MADNESS project).
As for the people that contributed to the success of this PhD, we have (drum roll):

- My advisor Paolo, who gave me the opportunity to work on such an interesting topic and guided me to the correct path. I really appreciated all the trust you put in me since the very beginning, when I didn't even trust myself.

- Giuseppe, one of the best people I have ever met. I owe you much more than you know and I hope someday to repay you, but for now, EBBE'ECCO!

- All the other people at EOLAB, that thought me an amazing language. A language that has nothing to do with programming, computers or electronics, a language that more than any other reminds me that we are ~~just~~ humans after all. PS: scallate!

- The NetS$^3$ Lab group at the Italian Institute of Technology, especially Gian Nicola and Fabio. It was ~~fantastic~~ fotonico sharing this project with you, I hope we will have another chance of working together in the future.

- The 3·Brain team, that made me feel one of them since the very first day I arrived in Switzerland. Special thanks to Kilian, with the promise to kick your ass one day.

- All the other PhD students and incredible people met during these years. If you are reading this, I want to let you know that you were an invaluable part of this experience.

- My family, asking me what the project was about 200 times was probably quite helpful after all.

- Aurora's family, that has been like a second family for me during these years.

- All the other friends, despite which I was still able to finish the PhD.

- Aurora, that walked everyday on my side. I hope that we can take a lot more steps together, despite all the effort and against all the difficulties. I have been looking for the right words for a long time, but I'm still unsatisfied, so I have to use the help of a real philologist:
  *'It would be the death of you to come with me, Sam,' said Frodo, 'and I could not have borne that.'*
  *'Not as certain as being left behind,' said Sam.*
  *'But I am going to Mordor.'*
  *'I know that well enough, Mr. Frodo. Of course you are. And I'm coming with you.'*
  ...
  *'So all my plan is spoilt!' said Frodo. 'It is no good trying to escape you. But I am glad, Sam. I cannot tell you how glad. Come along! It is plain that we are meant to be together. We will go, and may the others find a safe road! Strider will look after them. I don't suppose we shall see them again.'*
  *'Yet we may, Mr. Frodo. We may,' said Sam.*
  John Ronald Reuel Tolkien - The Lord of the Rings

# Abstract

High-density microelectrode arrays (HDMEAs) feature thousands of recording electrodes in a single chip with an area of few square millimeters. The obtained electrode density is comparable and even higher than the typical density of neuronal cells in cortical cultures. Commercially available HDMEA-based acquisition systems are able to record the neural activity from the whole array at the same time with submillisecond resolution. These devices are a very promising tool and are increasingly used in neuroscience to tackle fundamental questions regarding the complex dynamics of neural networks. Even if electrical or optical stimulation is generally an available feature of such systems, they lack the capability of creating a closed-loop between the biological neural activity and the artificial system. Stimuli are usually sent in an open-loop manner, thus violating the inherent working basis of neural circuits that in nature are constantly reacting to the external environment. This forbids to unravel the real mechanisms behind the behavior of neural networks.

The primary objective of this PhD work is to overcome such limitation by creating a fully-reconfigurable processing system capable of providing real-time feedback to the ongoing neural activity recorded with HDMEA platforms. The potentiality of modern heterogeneous FPGAs has been exploited to realize the system. In particular, the Xilinx Zynq All Programmable System on Chip (APSoC) has been used. The device features reconfigurable logic, specialized hardwired blocks, and a dual-core ARM-based processor; the synergy of these components allows to achieve high elaboration performances while maintaining a high level of flexibility and adaptivity. The developed system has been embedded in an acquisition and stimulation setup featuring the following platforms:

- 3·Brain BioCam X, a state-of-the-art HDMEA-based acquisition platform capable of recording in parallel from 4096 electrodes at 18 kHz per electrode.
- PlexStim™ Electrical Stimulator System, able to generate electrical stimuli with custom waveforms to 16 different output channels.

- Texas Instruments DLP® LightCrafter™ Evaluation Module, capable of projecting 608x684 pixels images with a refresh rate of 60 Hz; it holds the function of optical stimulation.

All the features of the system, such as band-pass filtering and spike detection of all the recorded channels, have been validated by means of *ex vivo* experiments. Very low-latency has been achieved while processing the whole input data stream in real-time. In the case of electrical stimulation the total latency is below 2 ms; when optical stimuli are needed, instead, the total latency is a little higher, being 21 ms in the worst case.

The final setup is ready to be used to infer cellular properties by means of closed-loop experiments. As a proof of this concept, it has been successfully used for the clustering and classification of retinal ganglion cells (RGCs) in mice retina. For this experiment, the light-evoked spikes from thousands of RGCs have been correctly recorded and analyzed in real-time. Around 90% of the total clusters have been classified as ON- or OFF-type cells.

In addition to the closed-loop system, a denoising prototype has been developed. The main idea is to exploit oversampling techniques to reduce the thermal noise recorded by HDMEA-based acquisition systems. The prototype is capable of processing in real-time all the input signals from the BioCam X, and it is currently being tested to evaluate the performance in terms of signal-to-noise-ratio improvement.

# Table of contents

# List of figures

# List of tables

# Nomenclature

**Greek Symbols**

| | |
|---|---|
| $\delta$ | Kronecker delta function |
| $\theta$ | Heaviside step function |

**Other Symbols**

| | |
|---|---|
| $\sigma_n$ | Estimated standard deviation of the noise |
| $f_s$ | Sampling Frequency per electrode |

**Acronyms / Abbreviations**

| | |
|---|---|
| *ADC* | Analog-to-Digital Converter |
| *AMP* | Asymmetric Multi Processing |
| *AP* | Action Potentials |
| *API* | Application Program Interface |
| *APS* | Active Pixel Sensor |
| *APSoC* | All Programmable System on Chip |
| *ASIC* | Application-Specific Integrated Circuit |
| *AXI* | Advanced eXtensible Interface |
| *BCI* | Brain-Computer Interface |
| *BMI* | Brain-Machine Interface |
| *BRAM* | Block RAM |

| | |
|---|---|
| *CLB* | Configurable Logic Block |
| *CMOS* | Complementary Metal–Oxide–Semiconductor |
| *CPLD* | Complex Programmable Logic Device |
| *CPU* | Central Processing Unit |
| *DDR* | Double Data Rate |
| *DLP* | Digital Light Processing |
| *DMA* | Direct Memory Access |
| *DNI* | Direct Neural Interface |
| *DSE* | Design Space Exploration |
| *DSP* | Digital Signal Processor |
| *DWT* | Discrete Wavelet Transofrm |
| *EEG* | ElectroEncephaloGraphy |
| $EEPROM/E^{2}PROM$ | Electrically Erasable Programmable Read-Only Memory |
| *EPROM* | Erasable Programmable Read-Only Memory |
| *FET* | Field Effect Transistor |
| *FFT* | Fast Fourier Transform |
| *FIR* | Finite Impulse Response |
| *FPGA* | Field Programmable Gate Array |
| *FPROM* | Field Programmable Read-Only Memory |
| *GPU* | Graphical Processing Unit |
| *HDL* | Hardware Description Language |
| *HDMEA* | High-Density MultiElectrode Array / High-Density MicroElectrode Array |
| *HDMI* | High-Definition Multimedia Interface |

| | |
|---|---|
| *IC* | Integrated Circuit |
| *IFR* | Instantaneous Firing Rate |
| *IIR* | Infinite Impulse Response |
| *LAN* | Local Area Network |
| *LE* | Logic Element |
| *LFP* | Local Field Potentials |
| *LSB* | Least Significant Bit |
| *LUT* | LookUp Table |
| *MAC* | Multiply and ACcumulate |
| *MEA* | MultiElectrode Array / MicroElectrode Array |
| *MFR* | Mean Firing Rate |
| *MMI* | Mind-Machine Interface |
| *NCI* | Neural-Control Interface |
| *OS* | Operating System |
| *OSR* | OverSampling Ratio |
| *PAL* | Programmable Array Logic |
| *PID* | Proportional Integral Derivative |
| *PLA* | Programmable Logic Array |
| *PLD* | Programmable Logic Device |
| *PROM* | Programmable Read-Only Memory |
| *RAM* | Random Access Memory |
| *RGC* | Retinal Ganglion Cell |
| *RISC* | Reduced Instruction Set Computer |

| | |
|---|---|
| *RMS* | Root Mean Square |
| *SD* | Secure Digital |
| *SEM* | Standard Error of the Mean |
| *SNR* | Signal-to-Noise Ratio |
| *SoC* | System on Chip |
| *TTL* | Transistor-Transistor Logic |

# Chapter 1

# Context and Motivation

*"The Hitchhiker's Guide to the Galaxy is quite positive I think.
It highlighted an important point, which is that a lot of times the question is harder than the answer and if you can properly phrase the question, then the answer is the easy part."*
Elon Musk (CEO of SpaceX, Tesla, and Neuralink)

## 1.1 You won't run away no more, I promise

Joseph Jérôme Lefrançois de Lalande is mainly famous for his scientific works on astronomy. Among those works, the first accurate measure of the distance between the earth and the moon, that he did when he was only 20 years old. However, more interesting about him is that he used to fill a snuffbox with living spiders for a teaching purpose. When the sky was clear, he used to stand outside with a little telescope and teach people about astronomy. However, many times the people just overstepped him without any care about what was in the sky. In those cases, he used to take out the snuffbox from his pocket, and eat a living spider in front of everyone. At this point, a large group of people would gather around to see what was going on, and Lalande who had just gained his public attention could start to teach about the wonders of astronomy. The snuffbox filled with living spiders is basically a trick to attract people attention to a subject that they would otherwise consider extremely boring and ignore completely. Now, my attempt is to use this story as Lalande's snuffbox to gain your attention hoping that you keep reading the next pages, that I assure you are far from being boring. The idea to use this story to gather people's attention has been clearly copied from (Bianucci, 2008).

The primary goal of this chapter is to point out what is the purpose of my PhD, and what is the motivation that pushed me in this research. This can be summarized as trying to answer the following question:

> **Can All-Programmable Systems on Chip be used in electrophysiology to study the dynamics of neural circuits by means of high-density microelectrode arrays, creating a real-time closed-loop neural interface?**

For those not fully familiar with the subject the question may seem a little bit obscure. The lack of vocabulary is probably one of the reasons, as the journalist Henry Hazlitt once said: "if you do not know the words, you can hardly know the thing." For this reason, in the following sections of this chapter, before explaining the question and how I came up with it, I will give a brief introduction to the topics of this research. In this way, I hope that everyone can build up the necessary vocabulary to understand this thesis and the work behind it.
The chapter is organized as follows: section 1.2 gives an introduction to electrophysiology and its history, the next section, 1.3, goes more into detail about neural circuits, the basic aggregates responsible of any function in the nervous system. Section 1.4 deals with HDMEA, one of the main devices used nowadays to acquire information from neural circuits; this brings us to the section 1.5 about neural interfaces, used to connect a biological neural circuit with an artificial one. Finally, in section 1.6 we discuss the All-Programmable System on Chip, and why I decided to pick them for this PhD research.

## 1.2 Electrophysiology

Our journey begins more than 2 hundreds years ago at the University of Bologna, when Luigi Galvani started his famous experiments with electricity and frogs published in the 1791 (Galvani, 1791). Galvani observed that contractions can be stimulated in dead frogs' legs by connecting the nerves and the muscles with metallic conductors to form a sort of circuit. He believed that the muscles themselves contained electricity, that he called *animal electricity*. When this electricity was enabled to flow (through the circuit), then the muscles contracted. Fascinated by the work of Galvani, Alessandro Volta started his own experiments on the subject. He soon discovered that by connecting two points of the same nerve with two different metals was possible to produce contractions in the frog's legs. At this point, he supposed that they were not the muscles containing electricity, but it was rather generated somehow by the connection of two different metals, and that the muscles were simply contracting as a response to an external stimulus. Continuing his research in this direction, Volta invented the electrical battery, which produced great excitement, and in the meanwhile, the experiments about frogs, muscles, and nerves were practically forgotten by most of the scientific community for the next decades. In any case, the Galvani experiments marked the beginning of electrophysiology (Piccolino, 1997), and time gave him reason in regard to the *animal electricity*.

> **Electrophysiology**
> (Oxford English Dictionary, 2018): A branch of physiology that deals with the electrical phenomena associated with physiological processes, esp. in nerve and muscle.
> (Scanziani and Häusser, 2009): By using metal, glass or silicon electrodes to record electrical signals associated with ion fluxes across neuronal membranes, electrophysiology allows us to listen directly to the 'language' of neurons at an extremely high signal-to-noise ratio.

One of the few who immediately continued the studies in this field was Galvani's nephew, Giovanni Aldini. At the beginning of 1800, thanks to the lack of ethical regulations, Aldini was able to conduct experiments, similar to those performed by Galvani, but instead of frog's legs, he used body parts of criminals previously put to death. As in the frogs, the human limbs contracted when current flowed through them. Inspired by his work, another young scientist Victor Frankenstein focused all his efforts to give back life to dead bodies with the use of electricity. The machine he developed was able to create a powerful electrical current through a human body. For his experiment, he assembled his *creature* with body parts from different corpses and positioned it in the machine. With this procedure, he was finally able to bring to life his creature (Shelley, 2009). Probably there is no need to specify that the Frankenstein's story is a fiction novel, but it is worth to know that the book was first published at that time, and it was truly inspired by the work of Aldini. This to say how strong was, even at those times, the influence that this kind of experiments was arousing on the general public. Horror fiction aside, Aldini gave a significant contribution to the field. He was the first to experiment the injection of electrical current to mammalian brains. After obtaining motor response in a ox by stimulating its brain, he thought of using galvanism to treat various diseases.

> **Galvanism**
> (Oxford English Dictionary, 2018): Electricity developed by chemical action. Also, the application of this for therapeutic purposes.

To properly understand the effects of electrical stimulation in the brain, he well though to apply galvanism to his own head as he described in (Aldini, 1804):
*"Je m'étais assuré sur moi-même, par l'application de l'arc sur toutes les parties de la face et de la tête, et par une foule d'expériences galvaniques , variées de toutes les manières, de l'influence énergique de ce stimulus sur l'organe encéphalique. En conséquence, j'ai appliqué un des conducteurs à une de mes oreilles, et l'autre, tantôt au nez, tantôt au front, de sorte que la tête fît partie de la chaîne qui conduisait l'influence galvanique, de la base au sommet de la pile. D'abord le fluide s'empara d'une grande partie du cerveau, qui éprouva une forte secousse, et comme une espèce d'ébranlement contre les parois de la boite osseuse. Les effets augmentèrent encore, lors que je conduisis les arcs d'une oreille à l'autre. J'ai ressenti une forte action à la tête, et une insomnie prolongée pendant plusieurs jours...".*
A true scientist who dedicated his body and soul to science. Thanks to his discoveries, Aldini was able to treat with electricity various patients from various kinds of mental diseases with discrete success. This was the origin of the modern electrotherapies that are in use today (Parent, 2004). A few years later, another Italian scientist, Leopoldo Nobili, was the first to record what Galvani called *animal electricity*, but he misinterpreted the results. He concluded that the electricity measured from the frog legs preparation was simply a thermoelectric current, produced by the different temperature of the two points. We have to wait until Carlo Matteucci for the correct interpretation: the current was generated by the living tissue (Verkhratsky et al., 2006). In the next years, the recordings from biological tissues continued, and more advanced techniques were developed in order to extract more accurate data. In 1949 Gilbert Ling and Ralf Gerard published a paper (Ling and Gerard, 1949) where they introduced the microelectrode as a tool to study the cell membrane potential in the muscle fiber of frogs. The microelectrode became suddenly the most used technique for electrophysiology.

> **Microelectrode**
> (Oxford English Dictionary, 2018): A very fine electrode, esp. one suitable for investigating the electrical activity of individual cells.
> (Ling and Gerard, 1949): We have pushed in the direction of drawing and filling microelectrodes of well under 1 μm tip-diameter and properly tapered and were rewarded by obtaining highly constant membrane potentials.

The improved version of the single microelectrode is the array of microelectrodes, also called multielectrode array (MEA). One of the first descriptions appeared in 1970, where an array of gold electrodes is placed on a silicon carrier (Wise et al., 1970).

> **Multielectrode Arrays (MEAs)**
>
> (Abee et al., 2012): Devices that contain multiple plates or shanks through which neural signals are obtained or delivered.

In 1972, instead, it was developed the first planar MEA to record from cultured cells (Thomas Jr et al., 1972).

> **Planar MEAs**
>
> (Franke et al., 2012): Two-dimensional arrangements of recording electrodes for *in vitro* extracellular measurements of cultured neuronal cells or slice preparations. They allow the recording of electrical activity simultaneously on many electrodes at high temporal resolution. Thus, they represent an important tool to study the dynamics in neuronal networks.

The schematic of one typology of MEA, the Utah array, is depicted in Fig. 1.1. The device features 100 three-dimensional microelectrodes on a matrix 10x10, and it is used for *in vivo* recordings.

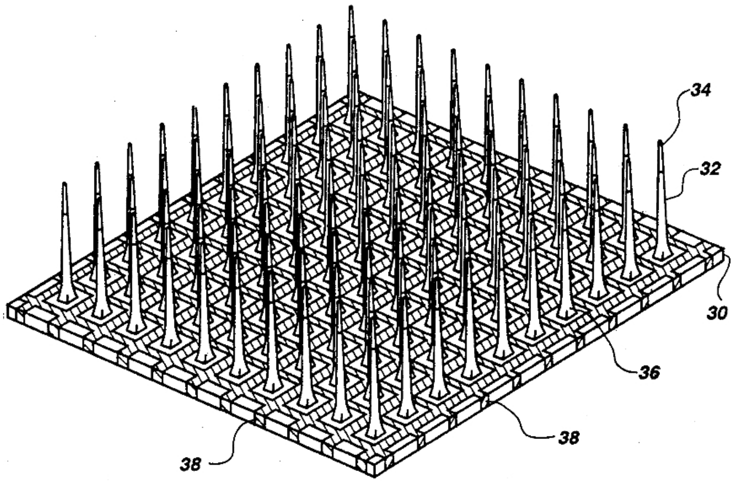


Figure 1.1 Schematic of a specific typology of MEA, the Utah array (Normann et al., 1993).

The full history of microelectrodes and MEAs is long and complex, and it is beyond the purpose of this thesis to give adequate coverage. Additional information about the most recent technology of MEAs will be given in the next chapters, for those interested to go deeper on the subject I suggest as a starting point the following sources (Cheung, 2007; Pine, 2006). The main advantage of using MEAs for electrophysiology studies is the possibility to record simultaneously from many different points in the biological tissue, this allows to study the electrical interaction among many different cells. Why is this so important? The answer lies in the next section.

## 1.3 Neural Circuits

It is a well-established knowledge in neuroscience that even the most basic neural functions are performed by fairly complex assemblies of single neurons. Eric Kandel (Nobel Prize in Physiology or Medicine, 2000), in his famous book *Principles of neural science* (Kandel et al., 2000), stated: *"The last frontier of the biological sciences – their ultimate challenge – is to understand the biological basis of consciousness and the mental processes by which we perceive, act, learn, and remember."* Now, in order to understand this biological basis, to unravel the mystery behind such complex phenomena that enable us to think, there is no other way but to study complex assemblies of neurons, since the study of the single cell cannot give the answer. But let's take a step back to better understand this. In 1949, Donald Olding Hebb, a Canadian psychologist, formulated his theory about cell-assemblies that he published in the now-famous book *Organization of Behavior* (Hebb, 1963). Hebb wrote: *“Any frequently repeated, particular stimulation will lead to the slow development of a "cell-assembly," a diffuse structure comprising cells in the cortex and diencephalon (and also, perhaps, in the basal ganglia of the cerebrum), capable of acting briefly as a closed system, delivering facilitation to other such systems and usually having a specific motor facilitation. [...] It is not necessary, and not possible, to define the cell-assembly underlying a perception as being made up of neurons all of which are active when the proper visual stimulation occurs. One can suppose that there would always be activity in some of the group of elements which are in functional parallel.”* What Hebb understood much before the others is that single neurons are not capable of pretty much anything on their own, but need to work together forming neural circuits. Moreover, he also proposed that a single neuron can be part of more than one neural circuit with a specific function.

> **Neural Circuits**
> (Purves et al., 2011): Neurons never function in isolation; they are organized into ensembles or neural circuits that process specific kinds of information and provide the foundation of sensation, perception and behavior.

This hypothesis was far ahead of his time, and the technology available was not advanced enough to investigate assemblies of many neurons, thus Hebb's theory was neglected for many years. The recent technological advancements in the electrophysiological techniques, such as the development of the MEA, made it possible to record the electrical activity from many densely integrated electrodes simultaneously. With this new tool, it is possible to record the neural activity of a vast assembly of cells with a good spatial resolution, thus allowing to study the interactions between the different neurons. The new evidence demonstrated that

Hebb was right, and his theory has been now highly revalued (Dalenoort, 1982; Nicolelis et al., 1997). Hebb is now considered as the father of neuropsychology and neural circuits.

> **Neuropsychology**
> (Oxford English Dictionary, 2018): The relationship between neural and mental or behavioural processes; a branch of psychology dealing with this.
> (Berlucchi, 2017): The discipline which investigates the relations between brain processes and mechanisms on one hand, and cognition and behavioral control on the other.

Neural circuits are also called neural networks, giving more emphasis to the topology of the biological system formed by highly interconnected nodes (neurons). However, if we look at the definition of neural network:

> **Neural Network**
> (Oxford English Dictionary, 2018): An interconnected system inspired by the arrangement of neurons in the nervous system; a program, configuration of microprocessors, etc., designed to simulate this.

This because, nowadays, the term is more used as referred to artificial systems that mimic the behavior of biological neural circuits. In this thesis, unless otherwise specified the term will always be used as referred to biological assembly of neurons. In the last decades, since Hebb's theory has been proved, neurobiology and neuroscience have witnessed a focus shift from the study of the single neuron to complex neural circuits.

> **Neurobiology**
> (Oxford English Dictionary, 2018): The biology of the nervous system; the branch of science dealing with this.

> **Neuroscience**
> (Oxford English Dictionary, 2018): Each of the sciences (as neuroanatomy, neurophysiology, etc.) concerned with the structure or function of the nervous system; such sciences collectively.

## 1.4 High-Density Microelectrode Array

As already anticipated, the standard device to study large neural assembly is the MEA. The processing of the signals acquired with such technology is not an easy task, for this reason, advanced processing systems are being developed every year. As it is described in (Franke et al., 2012; Obien et al., 2015) the new versions and architectures of these systems feature improvements in terms of:

- number of real-time processed parallel channels;
- supported sampling frequency;
- signal-to-noise ratio achieved;
- latency and accuracy of the real-time algorithms.

However, the technology of the acquisition platforms advances even faster. Looking at the history of the last half a century, we see that the number of the simultaneously recorded neurons is growing exponentially, doubling approximately every 7 years (Stevenson and Kording, 2011). The most recent versions of MEAs are the High-Density MEAs (HDMEAs) that contain thousands of densely integrated recording electrodes. Many commercially available HDMEA-based acquisition systems, produced for *in vitro* electrophysiology, can sample simultaneously around 1-4 thousand of electrodes, with a sampling frequency of 18–25 k Samples/s per channel and resolution of 10-14 bit, such as the 3·Brain BioCam X, the Multi Channel Systems CMOS-MEA5000-System or the Maxwell MaxOne Single-Well MEA. Moreover, up to 65 thousands electrodes in the same chip have been achieved (Tsai et al., 2017). Thanks to the high density of the electrodes, the HDMEAs allow recording from neural networks with a spatial resolution comparable to or even higher of that of the single cell (Obien et al., 2015). The signals coming from all these electrodes are sampled at the same time with a high frequency, in order to record all the details of the neural activity in the neural network (Seymour et al., 2017). Furthermore, many HDMEAs also include stimulating electrodes used to feed the neural circuit with electrical input with a time scale of microseconds (Obien et al., 2017). The need for an input to study the neural systems is mandatory if we really want to understand the working principles behind it. In fact, in nature, all of the neural circuits are constantly under the influence of some sort of input: the signals coming from the sensing organs. Those perturbations are what shape the neural network and make it behaves the way it does. For this reason, the neuroscientists started using stimulation for their studies. However, sending stimuli to the neural circuit without knowing the ongoing internal activity can cause very different responses. Indeed, the output of a dynamical system depends on the input, but also on the internal state. In a neural network, the internal state is rapidly changing, thus the need for a processing system that is able to follow these variations in real-time.

> **Real-time**
> (Oxford English Dictionary, 2018): Designating or relating to a system in which input data is processed so quickly so that it is available virtually immediately as feedback to the process from which it emanates.
> (Stankovic, 1988): In real-time computing the correctness of the system depends not only on the logical results of the computation but also on the time at which the results are produced.

Moreover, in nature, the biological neural systems are always interacting with the environment forming a sort of a closed-loop. The neural system in the animals makes them move in space responding to the stimuli received from the external environment, and thanks to their moving they receive different (probably better) stimuli.

> **Closed-loop**
> (Oxford English Dictionary, 2018): Characterized by a feed-back from a later to an earlier point in a cycle of operation.
> (G and Kriszta, 1985): Feedback (closed-loop) control is a strategy designed to achieve and maintain a desired process condition by measuring the process condition, comparing the measured condition with the desired condition, and initiating corrective action based on the difference between the desired and the actual condition.

In order to really understand the mechanisms behind the neural networks, open-loop stimulation is not enough, a closed-loop approach is needed in order to replicate the natural environment in which the neural networks normally behaves. The stimulation process must be tuned in accordance with the internal activity of the network and should evolve with it. This concept marked the beginning of the closed-loop neuroscience (Potter et al., 2014). It is relevant to point out that the technology resulted from this concept is used not only in laboratories to study the neural networks, but it has already reached practical applications. For example, restoring a spinal cord injury is not an easy task, bypassing the lesion with an artificial system is more feasible with the current technology (Lobel and Lee, 2014). Closed-loop neuroprostheses operate with the same concept, creating a bidirectional interface between the nervous system and the artificial limb. The feedback inserted in the loop improves the usability of the prosthesis, and thus the ability to perform tasks by the patients (Wright et al., 2016). Brain-controlled neurostimulation can also be used for therapeutic interventions to treat diseases such as epilepsy (Kassiri et al., 2017), or to speed up the recovery after an injury exploiting neural plasticity (Ethier et al., 2015). All the previously reported examples aim at the creation of a connection between a biological and an artificial circuit, what is called a neural interface.

## 1.5 Neural Interfaces

Actually, neural interface is just one of the many different names that exist to indicate the connection between the nervous systems with human made devices. All these names basically convey the same meaning with minimal differences: brain–machine interface (BMI), brain–computer interface (BCI), neural-control interface (NCI), mind-machine interface (MMI), direct neural interface (DNI).

> **Neural interfaces**
> (Krucoff et al., 2016): Neural- or brain-machine interfaces are electrode-computer constructs that extract and decode information from the nervous system to generate functional outputs.

When applied to the human body, neural interfaces can be divided into 2 main groups:

- Wearable: non-invasive devices, generally based on electroencephalogram (EEG) recordings, do not require any surgical operation but can be simply worn with almost full freedom of movement (Lin et al., 2009).

- Implantable: invasive devices, generally using microelectrode arrays surgically implanted, provide much higher bandwidth with the nervous system, with a cellular resolution unachievable with the wearable systems (Cheung, 2007).

More and more interest in neural interfaces is arising in the last years, especially in applications related to the human brain. This trend can be seen in the number of companies recently arising, and in the amount of money flowing in this brand new field. With no claim to be exhaustive, the Table 1.1 reports a small selection of recently founded companies with the aim of developing new technologies for neural interfaces. Some of these companies are founded by eminent entrepreneurs that put their own money on them and have set, as a final goal, the extension of the human cognition through implantable BMIs. Examples of these companies are NeuraLink founded by Elon Musk and Kernel founded by Bryan Johnson (Bryan with the 'y', not Brian Johnson, the singer of the AC/DC that messes with our brain in a totally different manner). Paradromics presents more or less the same goal, but is founded mainly by a government agency: the Defense Advanced Research Projects

Table 1.1 Investments in neural interfaces (source: crunchbase).

| Firm | Founded | Funding | Function |
|---|---|---|---|
| 3·Brain | 2011 | $1.8 M | High resolution MEA recording platforms |
| BrainCo | 2015 | $6 M | Wearable BMI for attention sensing |
| Dreem | 2014 | $57 M | Wearable BMI to enhance deep sleep |
| Kernel | 2016 | $100 M | Implantable BMI to treat diseases |
| MindMaze | 2012 | $110 M | Virtual reality to stimulate neural recovery |
| Neurable | 2015 | $2 M | Virtual reality controlled by wearable BMI |
| NeuraLink | 2016 | $27 M | High bandwidth implantable BMI |
| Paradromics | 2015 | $25 M | High bandwidth implantable BMI |
| SPR Therapeutics | 2010 | $44 M | Peripheral nerve stimulation to treat pain |
| Synchron, Inc. | 2016 | $10 M | Minimally invasive implantable BMI |

Agency (DARPA), part of the United States Department of Defense. SPR Therapeutics and Synchron, Inc. instead are solely focused on medical applications for now. SPR Therapeutics aims to reduce pain through peripheral nerve stimulation. Synchron, Inc. goal is instead to control advanced neural prosthesis through minimally invasive implanted BMI. Other companies, such as BrainCo and Dreem, focus on non-invasive wearable BMI to enhance brain functions: concentration, and deep sleep respectively. MindMaze and Neurable instead are about virtual reality applications of BMIs. The former offer rehabilitation programs based on virtual reality to stimulate neural recovery. The later instead is more focused on the entertainment sector, enabling people to play games using just their brain activity. Finally, the 3·Brain company is developing HDMEA biochips and recording platforms for *in vitro* electrophysiology. Some of the products developed by 3·Brain, and already commercially available, have been used in my thesis project; thus more details will be given later on in this document.

> **Biochips**
> (Oxford English Dictionary, 2018): Any of various microchip-like objects containing or consisting of biological elements, working in a biological environment, or combining some of these features within a single device; spec. (a) a silicon microchip inside a living organism; (b) a (hypothetical) logical device analogous to a silicon chip, but whose components are formed from biological molecules or structures; (c) a chip of silicon or other solid material on which microscopic arrays of molecules such as DNA sections or polypeptides are mounted for

simultaneous testing or analysis.
(Sohail and Li, 2018): A microarray (a collection of miniaturized test sites) arranged on a solid substrate that permits many simultaneous tests to be performed allowing higher-throughput volume and speed.

At this point, we have briefly reviewed the history of electrophysiology, we know what a neural circuit is and that the microelectrode array is one of the main tools used to study it. We pointed out the importance of real-time closed-loop neural interfaces, and the growing interest rising around them. However, there is one more thing to know about such closed-loop devices: they are extremely difficult to design, especially the processing part of the system. The 3 major problems to tackle are described in the following.

- Throughput: the number of parallel channels acquired is exponentially increasing, the analysis of such data in real-time is a very computation-intensive task (Maccione et al., 2015). In a state-of-the-art HDMEA acquisition system, several thousands of electrodes are sampled altogether at frequencies up to 25kHz, the produced data rate is very high, around 1 Gbps.

- Latency: in order to create a closed-loop, the response time of the processing system must be in the same time-scale of the fastest neural activity that we want to control, for action potentials this time latency must be maximum few ms (Buzsáki et al., 2012).

- Reconfigurability: the studies and applications in neuroscience evolve fast; the scientists need the ability to try out new experiments or new solutions as they discover something new. Moreover, the acquisition systems to record the signals from the cells are evolving too, to cope with this, some degree of reconfigurability is mandatory in the elaboration system to fit the different setups, applications, and experiments.

Nevertheless, analyzing the processing kernel for the realization of such systems, we can notice one peculiarity that can be exploited to facilitate the satisfaction of the required constraints: the same operations are repeated on every channel. A parallel processing architecture can be used to good advantage in order to achieve low latency and high throughput. If N different paths process the data from M different channels, then the throughput is N-times higher than what is achievable with a single path. Moreover, the samples from the last channels can be processed at the same time of the samples from the first channels, differently from what would happen in a sequential processing with a single path, where the first M-1 channels must be processed before going to the M$^{th}$; in this way the latency can be significantly reduced. For this work, among all the possible architectures that could suit the purpose, I have chosen the All-programmable System on Chip (APSoC).

## 1.6 All-Programmable System on Chips (APSoC)

An APSoC is basically a modern field programmable gate array (FPGA) with a bunch of additional hardware components all merged in a single chip. For those not familiar with FPGAs and APSoCs, I warmly suggest reading the Appendix A before proceeding with this section. Many of the information and details reported on the up mentioned appendix will be omitted here to avoid redundancy. Note that when there is no need to differentiate between FPGAs and APSoCs, the terms will be used interchangeably.

The main reason that pushed me into using this kind of devices for this work is that before starting the PhD, I already had quite some experience with FPGAs. As a main motivation, this may seem a bit weak, the point is that I knew the great advantages of its main characteristics for specific applications. The main features, programmability and parallelism, are perfectly suited to develop neural interfaces involving the parallel processing of huge amounts of data, while keeping the latency low and enough degree of reconfigurability:

- Programmability: an FPGA is a device that can be programmed on the field to become any logic circuit that you want. The specific category of FPGAs produced with the SRAM-based process can be reprogrammed theoretically infinite times. This makes it very easy to change the design after it has been developed, and considerably shorten the development time. Moreover, if the system to be implemented in the FPGA is well designed, with the right degree of parameterization and modularity, then it is straightforward to adapt it to different setups or to add new functionalities. This feature answers precisely the need of reconfigurability that we required in the previous section.

- Parallelism: the FPGA is composite of an array of configurable blocks independent one with the other, that can be connected to achieve complex functionalities. In this way, multiple parallel processing paths can be created in order to execute multiple tasks at the same time. As already anticipated before, this gives a boost to the achievable performances for the kind of application that we will implement on it. Moreover, in modern APSoCs, with integrated DSP slices and BRAMs, the available computing power, exploiting the parallelism, is much greater than most of the other processing architectures.

At this point of the discussion should be clear why I decided to try to use an APSoC as the processing element to build a neural interface. This is exactly the question that I started the chapter with, and that I report again here:

> **Can All-Programmable System on Chips be used in electrophysiology to study the dynamics of neural circuits by means of high-density microelectrode arrays creating a real-time closed-loop neural interface?**

Now the vocabulary should be enough for anyone to fully understand it, and of course, as the theory suggests and as this PhD work confirmed the answer is:

> **Yes.**

As a proof of what just said, in this thesis, I will describe the processing system developed in these three years of PhD, the experimental setup built to test it, and finally the experiments that were possible to make with such setup. This work will demonstrate that APSoCs are the perfect target technology to build closed-loop systems targeting HDMEAs with a high number of channels. As of now, the reached performance with regard to latency, parallel processed channels and degree of reconfigurability are not achievable with any other processing architecture.

The dissertation is organized as follows:

Chapter 2 is obviously the state of the art in this field, all the closed-loop system for electrophysiology developed in the last years are described here, pointing out the strengths and weakness of each one.

Chapter 3 describe the main work of this PhD, the processing unit and the experimental setup. Exploiting the architecture of a Xilinx APSoC, the Zynq®-7020 SoC, the system is able to process in real-time biological signals coming from thousands of parallel channels. The acquisition system used, the BioCam X, is currently commercialized by 3·Brain for *in vitro* electrophysiology. It makes use of an HDMEA chip featuring 4096 recording electrodes in a surface of a few square millimeters that are sampled at the same time with a frequency of 18kHz each. The acquisition system thus achieves submillisecond and micrometric resolution of the biological process.

Chapter 4 deals about an additional feature developed and implemented on FPGA that can be applied to all HDMEA acquisition systems, and exploiting oversampling techniques aims at reducing the thermal noise in the incoming biological signals.

The results are reported in the chapters 5 and 6. The former summarizes the performance reached with the developed hardware. The processing system is able to extract the main

features from all the incoming data, and based on that, it generates stimuli creating a closed-loop control with a latency of less than 2 ms from the signal acquisition to the stimuli decision. The later instead present the results of the biological experiments targeting mice retina exposed to light stimuli controlled by the processing system.
Finally, in the last chapter, I summarize all the research contributions achieved and discuss about the future work that can be made to improve the current system.

# Chapter 2

# State of the art

*"Remember that there is nothing in being superior to some other man.*
*The true nobility lies in being superior to your own previous self."*
Walter Lorenzo Sheldon (Lecturer of the Ethical Society of St. Louis)

## 2.1 There is safety in numbers

"Defendit numerus" used to say Giovenale, I must agree with him, after all, in God we may trust, but all others must bring data. In most sciences, numbers prove the validity of a concept; moreover, they help to put things into perspective and facilitate the understanding of the matter. So, I hope you like numbers; this chapter will be just about them.

Let us start with some fancy, interesting ones: 170 billion cells form a human brain of 1.5 kg, of these cells around 86 billion are neurons (Azevedo et al., 2009). The best MEA acquisition systems now available can classify around 1700 neurons in a single recording (Tsai et al., 2017) but only in *in vitro* experiments. While many improvements have also been made for *in vivo* recordings (Buzsáki et al., 2015), the number of neurons that can be classified in such experiments hardly reaches the thousand. That is only 85.999999 billion neurons away from the whole brain; we are close.

The chapter is organized as follows: the next section is about the state of the art of the MEA acquisition systems, then, in section 2.3, I report the MEA-based closed-loop systems implemented in the last years. Finally, in the last section, the advantages of my approach are summarized.

## 2.2 HDMEA acquisition systems

Now it is time to evaluate more in details the performances reached by the MEA acquisition systems. The major players in the field are university laboratories or research institutions and, more recently, also private companies. Without going too much back in time, we will start with the introduction of the complementary metal–oxide–semiconductor (CMOS) technology in the microelectrode array fabrication that brought a remarkable revolution in the field. The number of simultaneously recorded electrodes in a given area drastically raised, it is the beginning of the high-density MEA (HDMEA) era, with acquisition systems capable of recording or stimulating practically any neuron in the chip area (Hierlemann et al., 2015).
Most of the pioneering work in implementing CMOS-HDMEAs was done in a collaboration between the University of Neuchâtel, the University of Genova and the Italian Institute of Technology where various systems have been developed. Some of the people working there subsequently founded the 3·Brain company, where the systems designed in the academic environment were significantly improved and led to the development of the Biocam X, that is one of the current top HDMEA acquisition systems in the market, of which we will talk more in detail later.
The Bio Engineering Laboratory of the Eidgenössische Technische Hochschule (ETH) Zürich is another laboratory with a significant background in the field. An interesting system developed there is described in (Frey et al., 2009, 2010). The system exploits the switch-matrix concept allowing to select a flexible subset of the available electrodes. A total of 126 electrodes can be selected at the same time and used for recording at 20 kHz and at 8 bit resolution, or otherwise for stimulation. The whole chip features 11011 electrodes in an area of (2.00 mm × 1.75 mm) with a density of $3146\,\text{electrodes}/\text{mm}^2$, in comparison consider that in $1\,\text{mm}^2$ of a typical cortical culture are present between 500 and 2500 cells (Imfeld et al., 2008). The input referred noise is very low, being only $2.4\,\mu\text{V}_{\text{rms}}$. With the same concept of switch-matrix, it is also the system described in (Viswam et al., 2017, 2016) and developed in the same lab. This time, we have much more electrodes, 59760, achieving an electrode density of $5489\,\text{electrodes}/\text{mm}^2$ with an active area of (4.48 mm × 2.43 mm). From the full array, 2048 electrodes can be used simultaneously to record action potentials, while at the same time 32 more channels can record the local field potentials. Also in this system the noise is quite low, $3.2\,\mu\text{V}_{\text{rms}}$. The recording specifics per electrode are 20 kHz of sampling frequency and 10 bit for the resolution. Another acquisition system developed at the Bio Engineering Laboratory is described in (Ballini et al., 2014). Subsequently, people from the lab founded the Maxwell Biosystems company, and the system with some improvements

is now sold as the MaxOne Single-Well MEA, described later in this section.

The highest number of electrodes and parallel recording channels is reached by a system developed at the Columbia university (Tsai et al., 2015, 2017). We have in a single chip 65536 electrodes, each can be used for either recording or stimulation. The full array recording can be performed at 10 kHz and a 12 bit resolution, with an input noise around 10 $\mu V_{rms}$. The recording area is (6.53 mm × 6.53 mm), and the relative density of electrodes is 1537 $mm^{-2}$.

Now let us see what the market offers. We have the following companies: 3·Brain, Maxwell Biosystems, Multi Channel Systems (MCS) and, more recently, also Imec developing acquisition platforms for *in vitro* electrophysiology; their systems will be described in the following.

The last commercially available 3·Brain acquisition system is the BioCam X (3·Brain , 2018). As already anticipated, the system is an improvement of what previously developed in the academic environment and described in (Berdondini et al., 2009; Imfeld et al., 2008; Maccione et al., 2013). Two different MEA probes can be used with the BioCam X: the HD-MEA Arena and the HD-MEA Stimulo. The former features 4096 recording electrodes in an area of (2.67 mm × 2.67 mm) with a density of 575 electrodes/$mm^2$. The latter instead implements 16 stimulating electrodes in addition to the recording ones, but this time in a larger sensing area of (5.12 mm × 5.12 mm), resulting in a lower density of 156 electrodes/$mm^2$. Thanks to the implementation of the active pixel sensor (APS) concept, the signals from all the recording electrodes can be acquired at the same time. With both chips we have a sampling frequency of 18 kHz and a 12 bit resolution. Even recording from all the electrodes, the overall input noise is still low, 11 $\mu V_{rms}$.

Multi Channel Systems top product is instead the CMOS-MEA5000-System (MCS, 2018), with characteristics similar to the BioCam X. Again two different MEA probes are usable, each featuring 4225 recording electrodes and 1024 stimulation sites. What changes is the sensing area, one chip being (1 mm × 1 mm) while the other (2 mm × 2 mm). The resulting electrode density is huge: 4225 electrodes/$mm^2$ and 1056 electrodes/$mm^2$ respectively. Again the full array can be recorded at the same time, the sampling frequency is 25 kHz per electrode, and the resolution is 14 bit. The drawback of having such a high density lies in the input noise. The exact value is not reported in the website or the datasheet of the product, but we can suppose it to be more or less the same estimated for the system described in (Bertotti et al., 2014). That system represents a preliminary implementation done at the Technische Universität of Berlin, that led to the CMOS-MEA5000-System. The total noise estimation reported is more than 86 $\mu V_{rms}$, more or less an order of magnitude higher than the previously presented systems.

The third company to be presented is the Maxwell Biosystem, their acquisition system, the

MaxOne Single-Well MEA (Maxwell Biosystems, 2018), is a little different from the other two presented before. As for the BioCam X and the CMOS-MEA5000-System, also this platform is the result of a preliminary work done in the academic field, in this case, the ETH Zürich, and published in (Ballini et al., 2014). The main difference from the other commercial systems presented before is that it features 26400 electrodes. With a sensing area of (3.85 mm × 2.10 mm) the obtained density is $3265\,\mathrm{electrodes/mm^2}$. However, from the whole array, only a small subset of 1024 electrodes can be selected and sampled at the same time, with 10 bit resolution and 20 kHz sampling frequency. Additionally, 32 electrodes can be used for stimulation. The system presents the best noise level among the commercial systems, with only $4.4\,\mu V_{rms}$.
In February 2018, at the International Solid-State Circuits Conference (ISSCC) in San Francisco, also Imec introduced its HDMEA acquisition system (Lopez et al., 2018a). Described more in detail in (Lopez et al., 2018b), the system features 16384 total electrodes arranged in 16 clusters of 1024 electrodes each. The different clusters are separated by 1020 µm, allowing multi-well analysis to accelerate drug screening. Each cluster or well has an active area of (0.48 mm × 0.48 mm), achieving an electrode density of $4444\,\mathrm{electrodes/mm^2}$. The number of simultaneously recordable channels is limited to 1024 with a 30 kHz and a 12 bit resolution. Simultaneously, 64 channels can be used for stimulation. The input noise is in the same order of the other systems, $12\,\mu V_{rms}$.
The characteristics of the systems presented above are summarized in table 2.1. Additional information on HDMEA-based acquisition systems can be found in (Obien et al., 2015, 2017; Seymour et al., 2017).

Table 2.1 HDMEA acquisition systems for *in vitro* electrophysiology.

| Reference | Electrodes | Area | Density | Input | Stimul | $f_{samp}$ | Resolution | Noise |
|---|---|---|---|---|---|---|---|---|
| | | [mm$^2$] | [mm$^{-2}$] | channels | channels | [kHz] | [bit] | [μV$_{rms}$] |
| Arena - 3·Brain (2018) | 4096 | $2.67 \times 2.67$ | 575 | 4096 | - | 18 | 12 | 11 |
| Stimulo - 3·Brain (2018) | 4096 | $5.12 \times 5.12$ | 156 | 4096 | 16 | 18 | 12 | 11 |
| A - MCS (2018) | 4225 | $1 \times 1$ | 4225 | 4225 | 1024 | 25 | 14 | 86 |
| B - MCS (2018) | 4225 | $2 \times 2$ | 1056 | 4225 | 1024 | 25 | 14 | 86 |
| Maxwell Biosystems (2018) | 26400 | $3.85 \times 2.10$ | 3265 | 1024 | 32 | 20 | 10 | 4.4 |
| Lopez et al. (2018b) | 16384 | $16\ (0.48 \times 0.48)$ | 4444 | 1024 | 64 | 30 | 10 | 12 |
| Tsai et al. (2017) | 65536 | $6.53 \times 6.53$ | 1537 | 65536 shared | | 10 | 12 | 10 |
| Viswam et al. (2017) | 59760 | $4.48 \times 2.43$ | 5489 | 2048 | 16 | 20 | 10 | 3.2 |
| Frey et al. (2010) | 11011 | $2.00 \times 1.75$ | 3146 | 126 shared | | 20 | 8 | 2.4 |

## 2.3 Closed-loop systems

We have seen that HDMEA-based acquisition systems have reached incredible performances, with densities so high to allow recording at cellular resolution. With many models commercially available, these tools are nowadays increasingly used in neuroscience to uncover the mysteries of biological neural networks. Even if stimulation circuitry is present in practically all the systems currently produced, they still lack an essential characteristic. The Achilles' heel, that severely limits the potentiality of these systems in this research field, is the absence of closed-loop capabilities. As already anticipated in the introduction, if we want to explore the real mechanisms behind the biological neural systems, it is mandatory to provide a feedback. The simple open-loop approach, with stimulation and recording, is not enough to fully control and study neural responses. For this reason, closed-loop neuroscience is taking hold (Potter et al., 2014). In the laboratories around the world, various closed-loop prototypes have been developed, exploiting all sort of available processing architectures as elaboration hardware to analyze the neural signals and decide the appropriate stimulus. From complex analog electronics to plain software running on a desktop PC, passing through integrated circuits (ICs), FPGAs and digital signal processors (DSPs). In this section, a review of significant closed-loop systems featuring microelectrode arrays will be presented.

### 2.3.1 PC-based systems

The easiest and fastest way to implement a closed-loop system is to code the algorithms and run them in a desktop PC connected with the acquisition/stimulation setup. This has been the favourite approach for many years for experiments both *in vitro* (Newman et al., 2013; Novellino et al., 2007; Wallach et al., 2011; Zrenner et al., 2010) and *in vivo* (Rolston et al., 2009, 2010; Venkatraman et al., 2009).
A very good example of a PC-based closed-loop system is described in (Novellino et al., 2007). The embodied electrophysiology approach is exploited creating a bidirectional connection between an *in vitro* cultured biological neural network and an artificial system. As a neural network they used dissociated cultures of cortical neurons from rat embryos, while the artificial system is a mobile robot, creating something often referred to as hybrot.

> **Hybrot**
> (Rolston et al., 2010): A hybrid of a living neuronal system and a robot. It is most often used to refer to a mobile robot controlled by a neuronal network maintained in vitro. With such artificially embodied in vitro networks, the experimenter

> has complete control of the inputs to a simplified nervous system. The term can also describe animals or people with neural interfaces to robotic limbs or other mechanical actuators.

The link interface with the neurons consists on 32 recording channels sampled at 10 kHz, and 8 stimulation channels capable of sending both current and voltage signals. The desktop PC is in charge of detecting the spikes from the neural data, and based on the firing rate, it controls the neuro-robotic system with a latency of 4 ms. Additional control, such as tracking the robot movement, is done with two more PCs.

In (Venkatraman et al., 2009), instead, the PC is in charge of deciding the micro stimuli to send to an awake rodent, as a response to the recorded neural activity. A commercial multi-electrode array, chronically implanted on the barrel cortex, secures a 16-channel connection for both recording and stimulation. Either action potentials or local field potentials can be used to trigger a stimulation within 15 ms. Additional features include the tracking of the rat whiskers, that can also be used to activate the stimulation.

(Zrenner et al., 2010) demonstrate that even the MathWorks Matlab and Simulink programming framework, running on a standard PC, can be used to create a real-time closed-loop system with excellent performances. Targeting a 60-electrode MEA, where all the electrodes can be used for either recording or stimulation, they validated the system targeting *ex vivo* developing cortical neuronal cultures. Stimuli were sent after detecting a specific property on a spike train. With a sampling frequency of 16 kHz, they achieved sub-millisecond stimulation resolution, while the latency to change the stimulation electrode and waveform was under 10 ms. This system was subsequently improved implementing a proportional integral derivative (PID) controller on the spike probability, as described in (Wallach et al., 2011).

The lack of low-cost tools for closed-loop neuroscience was addressed in (Rolston et al., 2009, 2010), with the development of NeuroRigther. With a total cost under US$10 000, this open-source system features 64 electrodes recorded with a frequency up to 30 kHz; moreover, each electrode can also be used for stimulation. Again the processing takes place in a PC, this time running a C# application. After digital filtering, the input signals, both the spikes or the local field potentials can be selected to trigger the stimulation, achieving a total latency under 5 ms. The system has been validated in *in vivo* experiments, targeting the dorsal hippocampus of behaving rats. The NeuroRigther system was subsequently enhanced, redesigning the desktop application and adding features to allow sophisticated closed-loop experiments (Newman et al., 2013). In addition, online spike sorting was implemented, which slightly increased the input-output latency to 7.1 ± 1.5 ms. This time the system was validated with *in vitro* experiments over long time scales.

### 2.3.2 FPGA-based systems

The use of general purpose PCs for processing introduces variable delays causing timing jitter in the closed-loop response of the system. Among the major delay sources, we have the following: the operating system running in the background, any user interaction, additional interrupts from different parts of the system, buffers, and resource sharing (Müller et al., 2013). A possible solution to avoid all of this is to use a different processing architecture such as an FPGA device. The drawback is that much more effort is needed to design and implement the closed-loop system, but then the obtained performance can be significantly better in terms of throughput and latency.

Some of the first works introducing the FPGAs for real-time processing of MEA-acquired signal consisted on a hybrid approach where the FPGA is joined to some other processing architecture such as a desktop PC (Hafizovic et al., 2007), a RISC (Imfeld et al., 2009) or a DSP (Biffi et al., 2010).

I start presenting the hybrid approach described in (Hafizovic et al., 2007), where an FPGA is used in synergy with a PC to achieve a low-latency closed-loop. The FPGA device is in charge of acquiring the data coming from the MEA chip, at 20 kHz and 8 bit resolution, filtering it, and finally detecting and discriminating significant events as the presence of a single spike or a burst of spikes. A total of 128 bidirectional electrodes are featured in the used CMOS-MEA, meaning that each can be used both to record and stimulate. The PC is relieved from processing all the input data, and takes the stimulation decision based on the events detected by the FPGA; this allows to initiate stimulation in less than 2 ms after detecting a triggering event.

Other hybrid approaches featuring an FPGA are described in (Imfeld et al., 2009) and (Biffi et al., 2010), even if real-time processing of MEA acquired signals is achieved, they do not present closed-loop capabilities. In the former approach, a discrete wavelet transform (DWT) is implemented on the FPGA. It performs spike detection and sorting in real-time from 256 input channels, simultaneously recorded with 12 bit resolution and 18.75 kHz each. An additional reduced instruction set computer (RISC) was featured in the system for the computationally light tasks. In the latter approach instead, the FPGA is in charge of spike detection using a dynamically updated threshold, while the sorting is performed by an additional DSP. The number of channels that can be processed in real-time with this system is 64, each sampled at 25 kHz. In both cases, no information about the required processing time, and thus resulting latency, was given.

The pure FPGA design presented in (Müller et al., 2013) achieves much better performances than all the previously presented systems. The closed-loop stimulation latency is less than

1 ms while detecting the action potentials in real-time from 126 parallel channels previously digitally filtered. The input channels are selected among the 11011 electrodes available in the integrated MEA and sampled at 20 kHz; in addition, 42 more electrodes can be selected as output channels. The validation of the system was performed *in vitro* with cultured networks of cortical neurons.

A more recent FPGA-based system with a similar number of processed channel is reported in (Park et al., 2017), with 128 channels at 32.5 kHz. Like the previous system, digital filtering and spike detection are performed, but, in addition, we have also spike sorting executed online. The data is acquired from *in vitro* cultured cells, and the stimulation is performed through 8 selectable channels; however, no specifications are given about the latency of the system.

### 2.3.3 Something-else-based systems

As already anticipated before, also other hardware architectures rather than PCs and FPGAs have been used to implement closed-loop systems.

In (Cong et al., 2014), most of the processing is handled by custom blocks in an integrated circuit (IC) in charge of performing a fast Fourier transform (FFT). Later, a microprocessor uses the output from the spectrum analysis for the closed-loop control. Each electrode, 32 in total, is bidirectional and can be used as input and output; the acquisition is performed at 33 kHz and 13 bit. With this system, a brain-machine interface targeting a non-human primate has been realized and used for validation.

A similar approach is adopted in (Angotzi et al., 2014). The paper presents a wireless system to be used in *in vivo* experiments targeting small behaving laboratory animals such as rats. The system is composed of 3 main parts: the Remote Unit (i.e., Headstage and Backpack carried by the animal), the Home Unit (the wireless receiver) and a desktop PC. The Remote Unit is composed of an application specific integrated circuit (ASIC) and a microcontroller; it records the neural signals from 8 electrodes at 10.4 kHz and 8 bit, elaborates them, and sends stimuli with an additional 8 electrodes. The Home Unit creates a wireless bi-directional interface between the Remote Unit and the PC. While operating in the local closed-loop mode, the system sends stimuli with 3 ms latency after the detection of a spike. In the remote closed-loop mode, instead, stimuli are sent with 2.6 ms after the reception of a (Transistor-Transistor Logic) TTL trigger generated by the behavior of the rat.

In (Liu et al., 2017), instead, the elaboration is performed through an analog circuit and targets experiments in behaving animals. Recording and stimulation are performed with 16 shared electrodes. The closed-loop is realized implementing a PID controller in each channel. The PID is based either on the neural energy of the input signals or on the action potential firing rate. The spectrum analysis is performed in the analog circuit, while the firing rate is calculated on an external microcontroller.

### 2.3.4 Optogenetics

To conclude this review of closed-loop systems, I will present here some works that exploit the optogenetics concept, i.e., optical stimulation.

> **Optogenetics**
> (Oxford English Dictionary, 2018): A technique in neuroscience in which genes for light-sensitive proteins are introduced into specific types of brain cell in order to monitor and control their activity precisely using light signals.
> (Grosenick et al., 2015): A methodology that allows millisecond-scale optical control of neural activity in defined cell types during animal behavior.

(Nguyen et al., 2014) present a software performing spike detection and sorting to control in closed-loop an headstage connected to awake rats. The software, developed in LabVIEW and Matlab, takes as input the signals from 32 electrodes sampled at 12.5 kHz and 16 bit. After the online elaboration of the signals, the software sends as a response the commands for the optical stimulation with a latency around 8 ms.
In (Liu et al., 2018), instead, the closed-loop is achieved with a microcontroller integrated into the system. Featuring 16 graphene microelectrodes, the system sends an output optical stimulus each time one of the recorded signals is above a set threshold. The system has been tested with an input sampling rate of 1 kHz, achieving 1 ms latency, but it can theoretically go at 30 kHz with a latency as low as 0.04 ms.
For those interested, a review of many other (older) optogenetics systems is provided in (Grosenick et al., 2015).

Table 2.2 summarizes the presented systems. If we compare it with the previous Table 2.1 of the available HDMEA-based acquisition systems we can note that the potential number of input channels is not exploited in any existing closed-loop systems. This means these systems can only target neural circuits with limited size or only a local part of a bigger circuit.

Table 2.2 Closed-loop and real-time neurointerfacing systems.
Abbreviations:
DF = Digital Filter, SD = Spike Detection, DT = Dynamic Threshold for the SD, LF = LFP Feature Extraction, SS = Spike Sorting, SA = Spectrum Analysis, PID = Proportional Integral Derivative Controller

| Reference | Hardware | Input channels | Stimul channels | $f_{samp}$ [kHz] | Resolution [bit] | Processing Tasks | CL Latency [ms] |
|---|---|---|---|---|---|---|---|
| **This work (2018)** | APSoC | 4096 | 16 E + Optical | 18 | 12 | DF + SD + DT | < 2 |
| Liu et al. (2018) | $\mu$Controller | 16 | Optical | 1 | - | SD | 1 |
| Liu et al. (2017) | Analog + $\mu$Controller | 16 shared | | - | - | SD + PID | - |
| Park et al. (2017) | FPGA | 128 | 8 | 32.5 | 16 | DF + SD + SS | - |
| Nguyen et al. (2014) | PC | 32 | Optical | 12.5 | 16 | DF + SD + SS | 8 |
| Angotzi et al. (2014) | ASIC + $\mu$Controller | 8 | 8 | 10.4 | 8 | SD | 3 |
| Cong et al. (2014) | IC + $\mu$Controller | 32 shared | | 33 | 13 | SA | - |
| Müller et al. (2013) | FPGA | 126 | 42 | 20 | 8 | DF + SD | <1 |
| Newman et al. (2013) | PC | 64 shared | | 25 | 16 | DF + SD + SS | 7.1 ± 1.5 |
| Wallach et al. (2011) | PC | 60 shared | | 16 | 12 | SD + PID | < 10 |
| Rolston et al. (2010) | PC | 64 shared | | 30 | 16 | DF + SD + LF | < 5 |
| Biffi et al. (2010) | FPGA + DSP | 64 | - | 25 | 12 | SD + DT + SS | No CL |
| Imfeld et al. (2009) | FPGA + RISC | 256 | - | 18.75 | 12 | SD + SS | No CL |
| Venkatraman et al. (2009) | PC | 16 shared | | - | - | DF + SD + LF | 15 |
| Hafizovic et al. (2007) | FPGA + PC | 128 shared | | 20 | 8 | DF+SD | < 2 |
| Novellino et al. (2007) | PC | 32 | 8 | 10 | - | SD | 4 |

## 2.4 «It’s over 4000!!!»

screamed Vegeta[1] astonished, noticing the number of channels elaborated in parallel in my closed-loop system. Indeed, to advance the state-of-the-art, overcoming the limitations of the previously presented works, in this PhD I developed a new processing system. The aim is to cope with the incredibly high data rates produced by the most recent acquisition systems featuring the CMOS-MEA technology. The target acquisition system chosen for the experimental setup is the 3·Brain BioCam X described in Section 2.2, featuring 4096 recording electrodes sampled at the same time. The developed system is capable of closing the loop processing in parallel all the signals coming from the HDMEA device. Different validation experiments have been performed targeting *ex vivo* mouse retinas, and all the functionalities of the system have been assessed.
Summarizing, the developed system:

- is the first to exploit a modern All-Programmable SoC device; taking advantage of the offered heterogeneous processing architecture, it is able to elaborate very high data rates in the programmable logic and hardwired blocks while keeping an excellent degree of flexibility thanks to the integrated processor;
- escalates the number of parallel processed channels in real-time by more than one order of magnitude compared to the existing systems, while keeping the closed-loop latency very low, under 2 ms;
- integrates both electrical and optical stimulation capabilities;
- features high configurability, making it easy to adapt to various experimental setups with different acquisition and stimulation systems.

[1] fictional character in the Dragon Ball manga series created by Akira Toriyama, prince of an extraterrestrial race of warriors known as the Saiyans.

# Chapter 3

# Assembling the system

*"Assembling is a work that everybody should study by himself, with his own head and even better with his own hands: because you know, the things, seeing them from an armchair or from a pylon forty meters tall, it is different."*
Translated from *La chiave a stella* by Primo Levi (chemist, writer, and Holocaust survivor)

## 3.1 From social to neural networking

We have finally reached the technical part, in this chapter I will present the developed system and its integration in an experimental setup. However, before starting the discussion on the technical work, I want to highlight here another essential point. This PhD marked the beginning of a tight collaboration between the Microelectronics and Bioengineering Lab (EOLAB) at the University of Cagliari and the NetS$^3$ Lab at the Italian Institute of Technology in Genoa. None of the work presented in the following would have been possible without this collaboration and the synergy created between the electronics know-how from Cagliari and the neurobiological expertise from Genoa. So, I hope for this collaboration to continue long after the end of my PhD and that many other "kids" like me will benefit and learn from this multidisciplinary environment.

The chapter is organized as follows: I will start from a global view of the setup components and their interconnection, particular emphasis will be given to explain how the closed-loop is achieved, in the end, the details about the hardware processing system, that is the core of this work, will be given. While most of the designing for the processing system has been done in Cagliari, the final experimental setup was assembled at the NetS$^3$ Lab in Genoa.

## 3.2 Setup Overview

The ultimate experimental setup assembled during this PhD is depicted in Fig. 3.1, as of now it is up and running and currently being used in closed-loop experiments. Three main parts compose the closed-loop: the Acquisition unit, the Zynq-based Online Processing unit (Zyon for brevity) and the Stimulation unit, while an additional Offline Processing unit is present for additional control.

### 3.2.1 Acquisition unit

As already said before, one of the goals of this PhD was to fill the gap between the performances offered by the recent CMOS-MEA acquisition platforms and the closed-loop systems. For this reason, state-of-the-art commercially available products from the 3·Brain company have been chosen for the Acquisition unit: the BioCam X, which is claimed to be one of the most advanced MEA platforms for in-vitro electrophysiology (3·Brain , 2018), and its CMOS-MEA planar chips, HD-MEA Arena and HD-MEA Stimulo. Both chips feature 4096 recording electrodes, in addition, the HD-MEA Stimulo also contains 16 stimulating electrodes, and must be used whenever electrical stimulation is needed. The HD-MEA Arena is instead adopted for all the other experiments, in those cases, the loop is closed through optical stimulation. The BioCam X is able to record from the whole array with a sampling

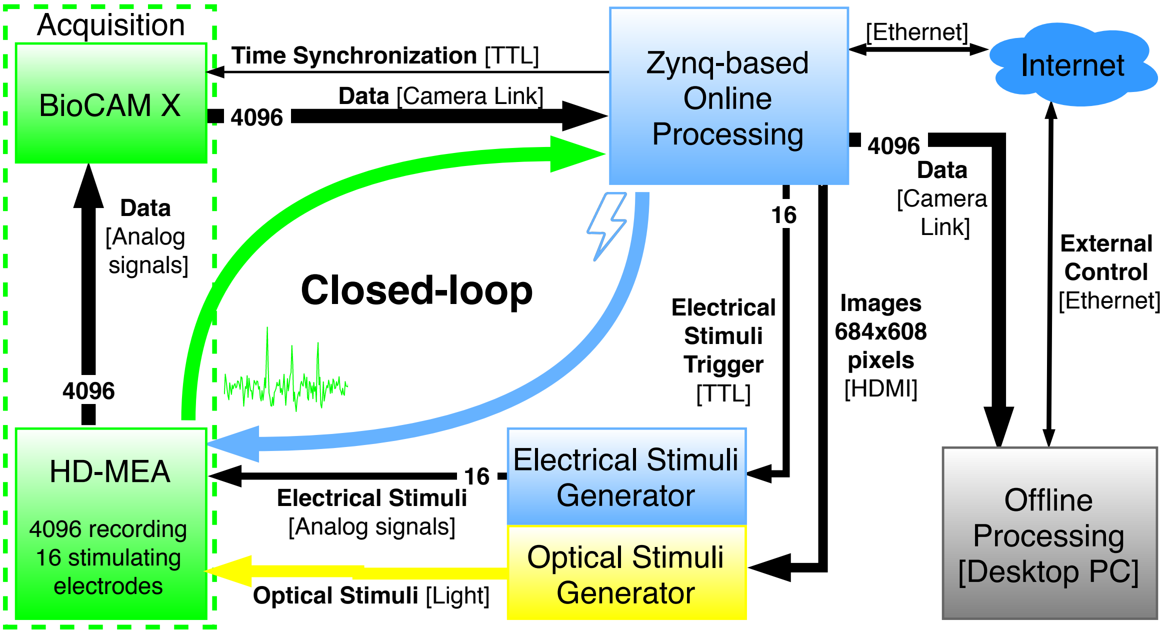

Acquisition
BioCAM X
Time Synchronization [TTL]
4096
Data [Camera Link]
Zynq-based Online Processing
[Ethernet]
Internet
4096
Data [Camera Link]
Data [Analog signals]
Closed-loop
16
Electrical Stimuli Trigger [TTL]
Images 684x608 pixels [HDMI]
External Control [Ethernet]
4096
HD-MEA
4096 recording 16 stimulating electrodes
16
Electrical Stimuli [Analog signals]
Electrical Stimuli Generator
Optical Stimuli Generator
Optical Stimuli [Light]
Offline Processing [Desktop PC]

Figure 3.1 Experimental setup schematic overview.

rate of 18 kHz per electrode and 12 bit resolution. All the recorded signals together with additional control bits are sent out through a Camera Link interface with a data-rate around 1 Gbit/s. In a standard open-loop setup the Camera Link would be connected directly to an acquisition board on a desktop PC where the data would be collected and elaborated. In this closed-loop setup instead, the Camera Link is connected to the Zyon.

### 3.2.2 Zynq-based Online Processing (Zyon) unit (brief description)

The Zyon is implemented on a ZedBoard™ development kit by Avnet featuring a Xilinx Zynq®-7020 All-Programmable SoC. This unit is in charge of multiple tasks:

- real-time processing all the incoming neural signals, in order to extract significant features that are meaningful for closed-loop experiments;
- running programmable closed-loop algorithms, deciding the feedback to be sent back to the biological neural network as a response of the detected neural activity;
- sending triggers for the Stimulation unit in the form of 16 TTL digital signals for electrical stimulation or 684x608 pixel images for optical stimulation;
- sending a copy of all the acquired data out through another Camera Link connection;
- running a Linux-based operating system with Internet connectivity with the purpose of interfacing the system to the external user, this allows to modify the HW/SW parameters remotely at any time adjusting the elaboration to the experiment;
- the Internet connectivity is also used to send the online results to the Offline Processing unit;
- keeping time consistency among the different units of the setup, thanks to the time synchronization signal.

Being the core of this PhD work, this unit will be described in detail in section 3.4.

### 3.2.3 Stimulation unit

Depending on the experiment and the cells under study, it can be either an Electrical or an Optical Stimuli Generator.

When electrical stimuli are needed, any platform able to receive TTL digital signals as triggers to provide preloaded analog stimuli is suitable for the purpose, either commercial or custom made. The particular platform integrated into the current setup is the Plexon PlexStim™ Electrical Stimulator System, that allows to send electrical stimuli with custom waveforms to a selected output channel (among the 16 available) upon detection of a TTL trigger in the correspondent digital input. The latency between the detection of the input trigger to the relative generation of the output signal is 1 µs, thus perfectly suitable for closed-loop applications. The electrical stimuli reach the biological tissues through the 16 stimulating electrodes of the HD-MEA Stimulo.

Other experiments instead involve optical stimulation, e.g., targeting the neurons of the retinas; in this case, the stimulus is a light pattern projected to the cells. In this particular setup the visual stimuli are provided by the Texas Instruments DLP® LightCrafter™ Evaluation Module. The module receives 608x684 pixels images through a High-Definition Multimedia Interface (HDMI) and projects them with a refresh rate of 60 Hz. Little effort would be needed to modify the Zyon in order to integrate a different Digital Light Processing (DLP) system in the setup, even with different frame size or refresh rate.

### 3.2.4 Offline Processing unit

This last module consists simply of a desktop workstation featuring an acquisition board with a Camera Link port. This unit does not strictly interact with the closed-loop, but it is used to remotely control all the parameters of the Zyon unit in order to adapt the setup and the processing to different experiments. The workstation also runs the 3·Brain BrainWave software which controls the acquisition board to receive the neural traces, and also sends commands to the BioCam X. BrainWave is also in charge of visualization of the raw data, received from the Camera Link in real-time, and storage for further more complex elaborations. Thanks to the timing reference shared by all the units, it is possible to compare the online results of the Zyon unit with the offline results of the desktop workstation, this provides a good validation mechanism of the online algorithms.

## 3.3 Closed-loop

There are many different ways to close the loop when dealing with biological neural networks; this can be seen as one of the reasons why this field is encountering such a rapid growth (Potter et al., 2014). The variety of approaches that can be used is certainly a strength of the subject; nevertheless, sometimes it makes the discourse abstruse, and who is not into the field can find it hard to understand how exactly the loop is supposed to work. To avoid any misunderstanding, before discussing the details of the Zyon unit, I will present here the functioning of the loop in the presented setup.

### 3.3.1 Forward data flow

Everything begins in the biological neural network under study. The neurons composing it communicate between them exchanging electrochemical signals, this is what makes of the neuronal assembly a network. These signals are sensed by the electrodes that form the HDMEA, where the neurons lay. These analog signals are first amplified to reduce the impact of future noise sources and then are digitalized by the BioCam X. From here the signals go to the Zynq-based Online Processing unit. Here, after the acquisition, the neural traces, containing all the extracellular activity, are band-pass filtered to separate the two main components: action potentials (APs), also called spikes, and local field potentials (LFPs). The APs are the fast components of the extracellular potential, created by the activity of single neurons, they occupy the spectral band between 300 Hz and 6 kHz. The LFPs are instead slow fluctuations caused by the concurrent activity of many neurons, and their band is limited under the 300 Hz (Gibson et al., 2012). In the current implementation, only the APs are considered for the following processing while the LFPs are discarded. However, the flow to target the slow fluctuation would be practically the same, hence with minimal modification, the system can also target experiments focused on LFPs. The next elaboration step is the spike detection; an amplitude threshold is used to spot peaks in the signals representing messages sent between neurons. The detected activity is finally used by custom algorithms to decide which stimuli to send back to the neural network.

### 3.3.2 Feedback data flow

The stimulation decision taken by the Zyon is communicated to the Stimulation unit, this is the beginning of the feedback. The Stimulation unit produces either electrical or optical stimuli depending on the experiment, and based on the instructions received. The electrical stimuli are basically analog signals with custom waveform; they are conveyed to the biological tissues by charge carriers, such as electrons, passing through the stimulating electrodes featured in the HDMEA. The optical stimuli are instead projected pictures, this time the stimulation is conveyed by photons that reach the cells of the retinas through the air. The color and brightness of each pixel composing the image determine the characteristics of the stimulus, i.e., the number and the wavelength of the photons reaching each particular neuron. The stimulation, either electrical or optical, changes the neural activity of the cells under study, which will be sensed by the recording electrodes, and we are back at the beginning of the loop.

## 3.4 Zynq-based Online Processing (Zyon) unit

It is now time to talk about the design details of the fundamental part of this work: the Zynq-based Online Processing unit implemented on the Zedboard. The detailed architecture of this unit is depicted in Fig. 3.2. The main hardware component integrated on the board is the Zynq®-7020 APSoC that can be split in two parts: the *ZYNQ Processing System* (not to be confused with Zyon) featuring a dual core ARM®-based processor, and the *Programmable Logic & Hardwired Blocks* (that we will also call FPGA) featuring DSPs, BRAMs and

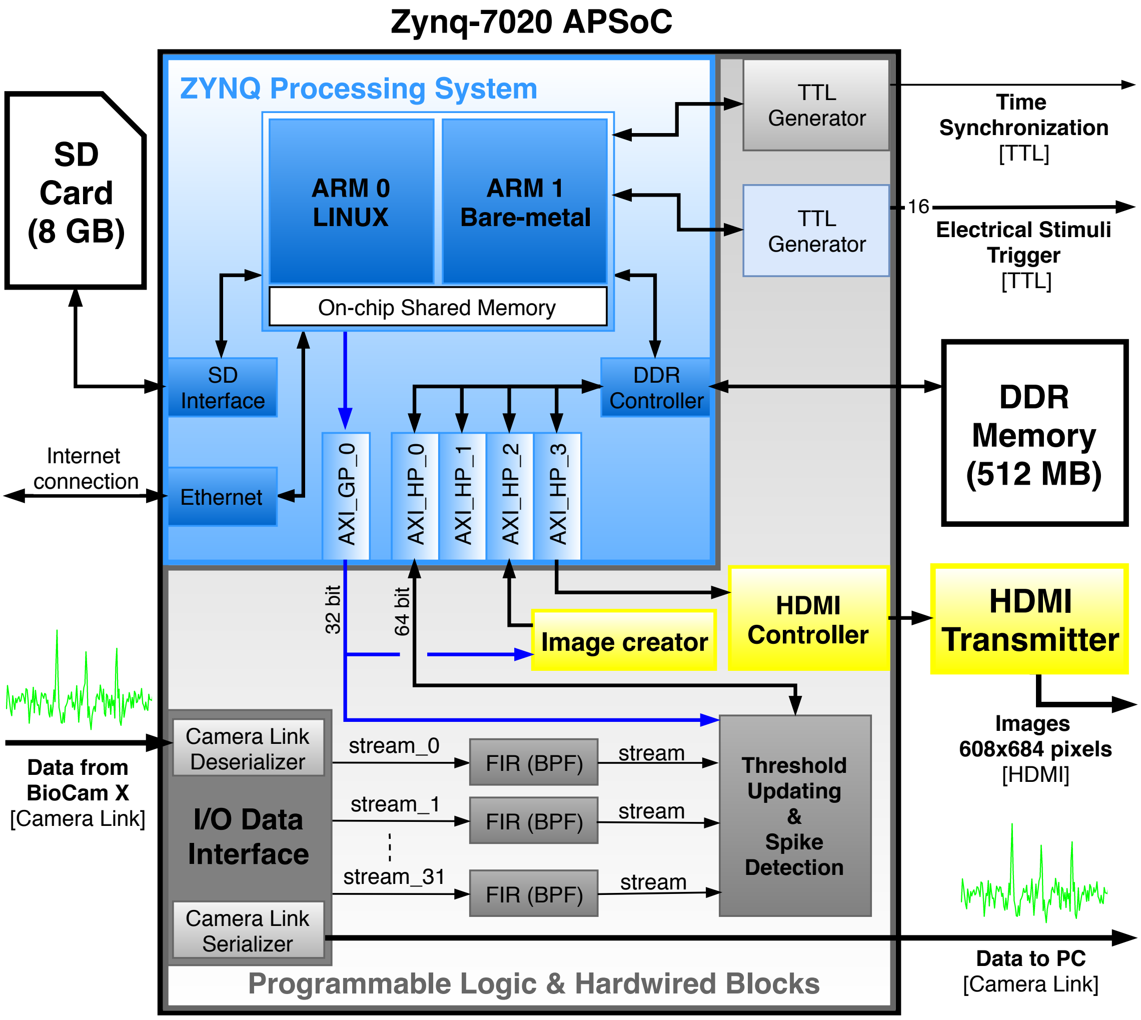


Figure 3.2 Zynq-based Online Processing (Zyon) unit.

LUTs. The tightly coupling and synergy between these two parts permit to achieve high throughput and low latency, thanks to the Hardware resources, while leaving a good degree of adaptability to the system, thanks to the software running on the processor. For more details about the APSoCs in general and the components featured on them see the Appendix A. Other hardware components featured on the Zedboard that have been exploited in the Zyon are the Secure Digital (SD) Card, the Double Data Rate (DDR) Memory, and the HDMI transmitter. In the following, we will have a closed view on each of these components.

### 3.4.1 Programmable Logic & Hardwired Blocks

This part is responsible for the most computationally intensive tasks of the processing chain, to achieve a high performance and flexibility a modular approach has been used. Multiple Intellectual Property (IP) cores, connected in a dataflow fashion, work in parallel to achieve the desired elaboration, as depicted in Fig. 3.2. The grey blocks in the bottom perform the first elaboration steps common to all the experiments; the yellow blocks are instead used only when optical stimulation is involved; finally, two TTL Generators are featured, one for the Time Synchronization signal and the other to trigger the electrical stimuli. In the following, each block will be described.

#### I/O Data Interface

It represents the interface of the FPGA with the Camera Link ports, one connected to the BioCam X and the other to the PC. Each Camera Link port is composed of four serial data links plus a synchronization clock. In this module, the input signals from the BioCam X are deserialized and interpreted based on the rules of the proprietary communication protocol adopted by 3·Brain, in this way the incoming samples are extracted and ready to be processed by the following modules. At each clock cycle of the synchronization clock a 28-bit data word is received: 4 bits are used as a control flag signaling when a word is valid and when a full frame of 4096 sample is received, the other 24 bits represent two samples recorded from two different electrodes (12-bit each). The detailed schematic of the Camera Link Deserializer is shown in Fig. 3.3. The serialization factor is 7, i.e., each clock cycle, 7 data beats occur; the goal of the block is to acquire all of them, from each line, and create the 28-bit output word. In order to do so a clock with 7× the synchronization clock frequency and phase aligned to it is needed. The Clock Manager is responsible for its creation; moreover, it produces the Receiver clock, which will be used by the next blocks to acquire the output word. The 7x clock is instead used by four Serdes (one per data line) to sample the input

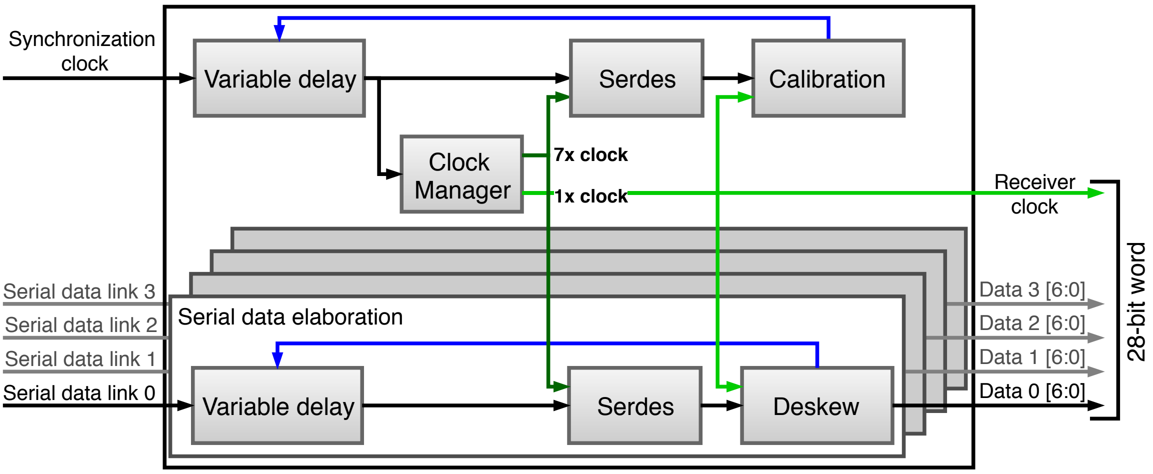


Figure 3.3 Schematic of the Camera Link Deserializer.

signals and produce 7-bit packets that will be assembled in one word. The frequencies in play are 50 MHz for the Synchronization and the Receiver clocks, and 350 MHz for the data links. To cope with the very fast data transitions, particular care must be given to the synchronization of the serial data links with the sampling clock to obtain an error-free data recovery. This is the purpose of all the additional blocks present in Fig. 3.3. The Calibration block calculates the time shift needed for the input clock line, while the Deskew modules calculate additional delay necessary to compensate the skew on each data line. The delays are adjusted at run-time thanks to the Variable Delay modules in order to perform the sampling of the data in the middle between two data transitions, i.e., where the data is more stable. The obtained 28-bit word is duplicated and sent to two different paths. One copy goes to the Camera Link Serializer, where it is converted back to its original form and sent out to the second Camera Link port connected to the desktop PC. Deserialization and re-serialization is a mandatory step to avoid violations of the timing in the programmable logic. From the other copy of the word, the 12 bit samples are extracted and fed to the next elaboration blocks. Since the data rate to elaborate is huge, 32 parallel paths are created for the next processing step; each path will process the signals from 128 electrodes. Hence the I/O Data Interface block is also in charge of routing the incoming samples correctly to the right stream interface.

### Band-Pass Filters (BPFs)

The 32 output stream interfaces from the previous block are connected to as many filtering blocks. As already anticipated the band-pass filtering is used to separate the spiking activity of single neurons from the local field potentials, they also remove all the unwanted noise that lays outside the spectrum of interest. In the current implementation the low and high cutoff frequencies of the designed filters are respectively 300 Hz and 3400 Hz. Each filtering block implements a Finite Impulse Response (FIR) filter that is able to process in a time division multiplexing fashion 128 different channels. The particular partitioning to reach the 4096 total channels to process, 32 blocks with 128 channels each, has been selected to minimize resources consumption while still allowing real-time processing, as will be detailed in the Chapter 5. The general schematic of a digital FIR is depicted in Fig.3.4, the output value $y_i$ is a sum of the most recent data samples, $x_i$ trough $x_{i-N}$, weighted by the coefficients $c_0$ trough $c_N$, where $N$ is called the filter order. Putting it as an equation:

$$y_i = \sum_{k=0}^{N} c_k \cdot x_{i-k} \tag{3.1}$$

The number of previous samples to be used to calculate the filtered value is the most important parameter to be decided by the designer. A higher filter order allows to design a more efficient frequency response with better selectivity of the desired spectrum; however, this is done at the cost of higher computational complexity and delay of the response. It is easy to see from the schematic that for a filter with order $N$, $N+1$ multiply and accumulate (MAC) operations are needed. The introduced delay instead is not so immediate to see, but can be easily calculated if we assume that the filter response presents a linear phase, which is the case. Linear phase implies that the group delay is constant for all the frequencies, and for the FIR is equal to

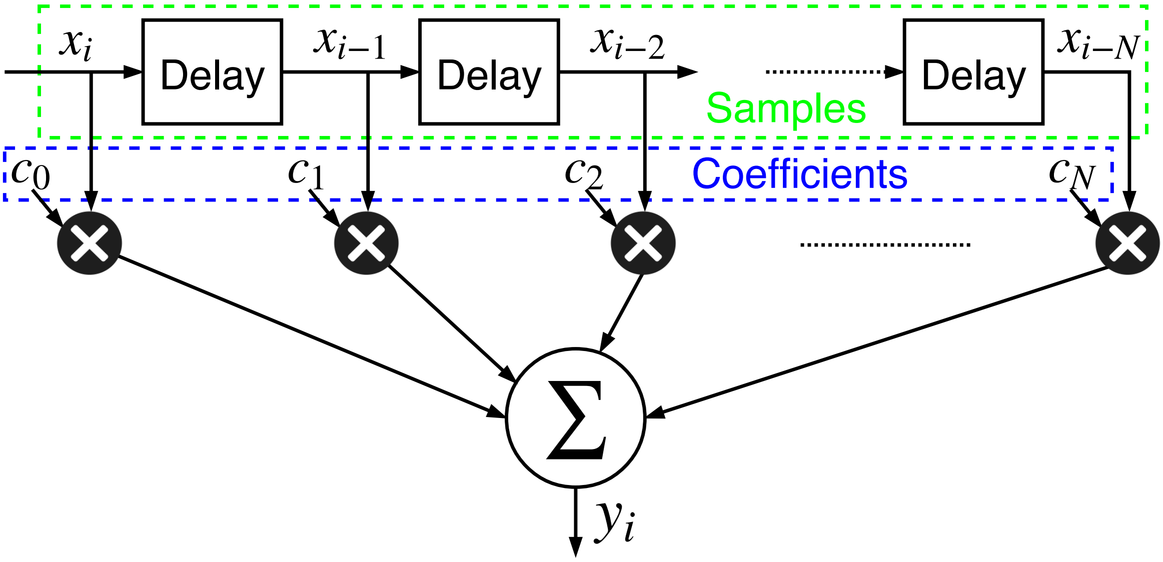


Figure 3.4 Finite Impulse Response filter schematic.

$N/(2 \cdot f_S)$, where $f_s$ is the sampling frequency of the incoming samples. Both the workload and the delay are directly proportional to the filter order; being it so critical, a question that I get often is why I picked FIR instead of Infinite Impulse Response (IIR) filters, which are known to achieve the same performance with a much lower order. The answer lays precisely in the linear phase response assumed before, that is mandatory in order to preserve the shape of the spikes in the signal. Indeed, any unwanted distortion would otherwise compromise all the subsequent processing. The linear phase response is straightforward to achieve with the FIR architecture by making the coefficient sequence symmetric. Instead, it is not possible to achieve such a response with IIR filters; thus additional complex equalizers would be needed to reduce the introduced distortion. It would be interesting to see if an optimal use of IIR filters + equalizers leads to better performances in terms of workload and latency, but this comparison was not featured in this work. As will be discussed later in Chapter 5, a filter order $N = 63$ has been chosen as the best trade-off between accuracy and computational performance. The total workload per second of the filtering phase can be now calculated as:

$$\begin{aligned} WL_{FIR} &= channels \cdot f_s \cdot \frac{(N+1)}{2}\,\mathrm{MAC} \\ &= 4096 \cdot 18\,\mathrm{kHz} \cdot \frac{(63+1)}{2}\,\mathrm{MAC} \approx 2.359 \cdot 10^9\,\mathrm{MAC/s} \end{aligned} \tag{3.2}$$

The division by 2 is due to the use of symmetric coefficients which halves the number of multiplications needed. Instead, the group delay introduced by the filtering phase is:

$$DELAY_{FIR} = \frac{N}{2 \cdot f_s} = \frac{63}{2 \cdot 18\,\mathrm{kHz}} = 1.75\,\mathrm{ms} \tag{3.3}$$

### Threshold Updating & Spike Detection

After the filtering phase, 32 stream interfaces (one from each FIR block) take the data to the next block, where the spike detection is performed. The common method described in the literature to detect the spikes involves the use of an amplitude threshold that is usually calculated as:

$$Thr = \alpha \cdot \sigma_n \tag{3.4}$$

where $\alpha$ is a constant usually between 3 and 5, and $\sigma_n$ is the estimated standard deviation of the noise (Gibson et al., 2012). The value of the threshold is critical to achieve accurate results. If it is set too high a lot of spikes will be missed, too low and noise peaks will be detected as well. Many solutions have been proposed to find an adequate value for the threshold automatically without the need for human intervention. An excellent example is described in (Quiroga et al., 2004), the threshold has been calculated with the equation 3.4 by setting $\alpha = 4$ and estimating automatically the noise standard deviation as:

$$\sigma_n = median\left\{\frac{|x|}{0.6745}\right\} \tag{3.5}$$

where x is the filtered neural signal. The use of the median allows to reduce the interference of the spikes in the estimation of the noise from the neural signal; indeed, the number of samples that are spikes are a small fraction of the total samples, thus they have little influence on the median. Another critical point to take into account, that can lead to substantial errors in the spike detection, is that the noise and signal levels are different from channel to channel, and moreover they can change over time. Some main factors that influence these levels are the following:

- distance of the cells from the recording electrodes;
- degradation state of the electrodes;
- behavior of the neuron;
- neural activity in the surrounding area.

In order to deal with it, the ideal solution would be to adjust the threshold dynamically and automatically considering only the most recent part of the signal. I obtained this by integrating a sliding window mechanism in the calculation of the threshold, performing an updated on the level with every new sample acquired. Combining the Quiroga approach of Equation 3.5 with the dynamic update of a different threshold for each channel is not practically feasible, due to the computational complexity needed to calculate the median. For this reason a lighter algorithm has been adopted, estimating the noise standard deviation as:

$$\sigma_n = \sqrt{\left(\sum_{k=0}^{M-1} x_{i-k}^2 - \left(\sum_{k=0}^{M-1} x_{i-k}\right)^2 \cdot \frac{1}{M}\right) \cdot \frac{1}{M}} \tag{3.6}$$

Here M is the size of the sliding window and $x_i$ the i-th filtered sample. Eq. 3.6 is actually the exact standard deviation of the last part of the whole signal (noise + spikes). The main downside of this approach is that high amplitude spikes or high firing rates in the signal could seriously alter the threshold value. To compensate this, the $\alpha$ in Equation 3.4 is dynamically changed between a preset range. The selected value of $\alpha$ depends on the difference between the peaks values on the signal and the standard deviation. Low computational complexity has been reached by adopting multiple hardware optimizations:

- choosing a windows size that is a power of 2, the divisions become simple bit-shifts;
- updating the sum terms only with the new acquired sample and the last in the window, instead of recalculating everything;
- avoiding the square root by using $\sigma^2$ to calculate a threshold square (and comparing it with squared samples already available).

The final processing flow for the spike detection is summarized in Fig. 3.5, it is easy to see that thanks to the optimizations only 4 MACs are needed for each cycle. The total workload is calculated in Equation 3.7

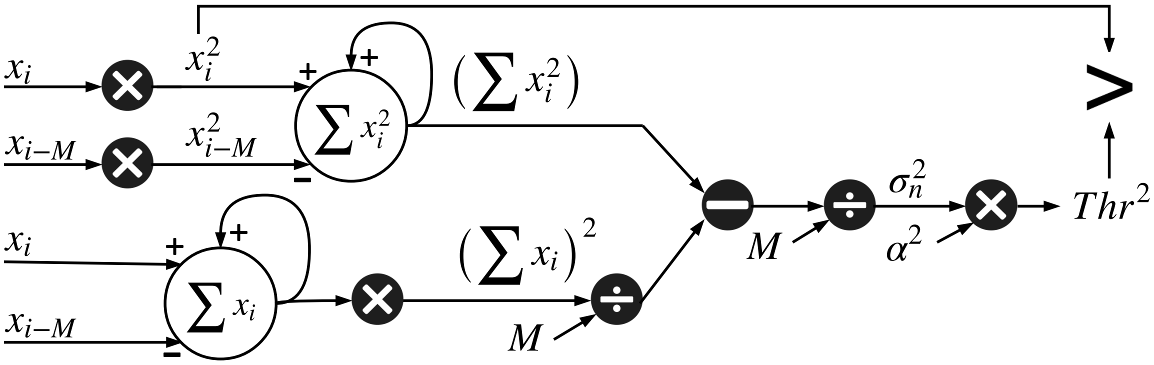


Figure 3.5 HW implementation of the spike detection algorithm.

$$\begin{aligned} WL_{detection} &= channels \cdot f_s \cdot 4\,\mathrm{MAC} \\ &= 4096 \cdot 18\,\mathrm{kHz} \cdot 4\,\mathrm{MAC} \approx 295 \cdot 10^6\,\mathrm{MAC/s} \end{aligned} \tag{3.7}$$

Note that, even if the size of the sliding window does not seem to directly affect the computational complexity, as we will discuss in a next chapter, it actually affects the resource occupancy. In the current implementation, a value of 4096 samples has been chosen corresponding to about 228 ms cause it was found to be the one providing the best detection performances, as will be detailed in Chapter 5.

An additional mode allows to use a static threshold instead of the dynamic one. A software running on the ARM processor is in charge of loading a different threshold value for each channel, any algorithm present in literature can be easily exploited in this mode. The resources usage impact of this second mode in the hardware is minimal, involving only simple

compare operations.
One last feature implemented in the module allows the user to specify the refractory time. After a spike has been detected in a particular channel, a hardware mechanism inhibits further spike detections in that channel for the specified amount of time. This is done to avoid the detection of the same spikes two times, and also to reduce the detection of false positives. Indeed, it reflects the actual biological mechanism of neurons: not able to fire spikes continuously, they must wait for the refractory period, usually around 2-3 ms (Rey et al., 2015), to be over before being able to release a new impulse.

A further discussion must be reserved now for the storage part. As can be seen in Fig. 3.2 the Threshold Updating & Spike Detection module is directly connected to the DDR controller through an Advanced eXtensible Interface (AXI) High-Performance port (AXI_HP_0), thus no processor intervention is needed to transfer data to and from the DDR. The module stores in the external memory both the filtered samples and the information about the detected spikes. For each frame, 4096 samples and 4096 flags are transferred to the DDR. The filtered samples must be saved to be later used for the dynamical update of the threshold ($x_{i-M}$ in Fig. 3.5), for this reason, the module also needs to read 4096 samples each step. The samples can also be used to calculate the static threshold in the ARM processor. The spikes information (time and channel where it has been detected) are instead used by the closed-loop software running on the ARM processor to decide the next stimuli. In future implementations, the software could also use the filtered samples to perform more advanced processing to close the loop. In order for the data to be easily accessible also from the processor, the sizes have been chosen as multiples of byte: the samples as 16-bit, and the flags containing the spike information as 8-bit. The total bandwidth for reading and writing can be then calculated as:

$$\begin{aligned} BW &= channels \cdot f_s \cdot (16+16+8)\,\mathrm{bit} \\ &= 4096 \cdot 18\,\mathrm{kHz} \cdot (16+16+8)\,\mathrm{bit} \approx 3\,\mathrm{Gbit/s} \end{aligned} \tag{3.8}$$

The bandwidth constraint is met thanks to the 64-bit wide bus that allows a transfer rate over 5 Gbit/s, considering the FPGA working frequency used, ~83 MHz. The specific system frequency has been chosen in order to achieve real-time processing (of all the incoming signals) while avoiding timing problems in the critical paths.

### Image Creator

This module is used only when optical stimulation is needed. It receives commands from the ARM processor and, as a response, writes in the DDR the images to be projected to the biological cells. The images are stored in the DDR as 32-bit RGB pixels; each frame is composed of 684x608 pixels, thus it occupies around 13 Mbit of memory. The module can change a whole frame at the same time or a subset of it, drawing geometrical shapes with one-pixel resolution. The images could have also be written in the DDR with the ARM core (in fact that was the first approach adopted), but having a dedicated hardware module presents multiple benefits. First, it frees the processor from this duty leaving more computational time for the other tasks, second, the creation of the images is much faster in hardware. Indeed, the hardware module writes the pixels in the DDR memory in burst mode, instead of establishing a new connection for each pixel with all the overhead implication, a single connection is established for all the pixels in a frame. Moreover, with a bus of 64-bit, two pixels can be transferred at the same time. The total time to write an entire frame in the DDR memory is thus:

$$DELAY_{full-image} = \frac{608 \cdot 684\,\mathrm{Pixels}}{2\,\mathrm{Pixels} \cdot 83\,\mathrm{MHz}} \approx 2.5\,\mathrm{ms} \tag{3.9}$$

### HDMI Controller

As the Image Creator, the HDMI controller is used only in experiments involving visual stimulation. It works like a Direct Memory Access (DMA) module, it reads autonomously from the DDR the images, previously written by the Image Creator, and send them to the HDMI Transmitter, that will control the projector. Differently from the Image Creator that writes only when commanded by the ARM processor, the HDMI Controller, once started, keeps reading continuously from the assigned location in DDR Memory. Two different ways are available to change the projected image:

- Overriding the memory data pointed by the module, ideal for small iterative changes of a small subset of pixels.

- Changing the settings of the module to point to a different address in the memory where the new image was previously written, ideal when the whole frame must be changed.

### TTL Generator

The last block to be described is the TTL Generator. Two instances of this module have been implemented in the Programmable Logic & Hardwired Blocks. The module allows to send TTL output signals of a very precise duration and with customed delay, ideal to send control commands to external components of the setup. The first instance of the module is used to produce the Time Synchronization signal that will be sampled by the BioCam X, and sent with the Camera Link interface together with all the biological data. This known signal acts as the timing reference among all the units of the setup, and it allows to compare the online results of the Zyon with the offline ones produced by the desktop PC representing an ideal mean for validation purposes. The second instance instead produces the triggers for predefined electrical stimulation. A total of 16 different channels can be used to trigger stimulations in different electrodes.

## 3.4.2 ZYNQ Processing System

The ZYNQ Processing System featured in the Zynq®-7020 APSoC (highlighted in blue in Fig. 3.2) is composite of a dual-core ARM® Cortex$^{TM}$-A9 processor clocked at 667 MHz and several hardwired controllers and interfaces. The two cores are exploited in an Asymmetric Multi-Processing (AMP) configuration with one core running a Linux-based operating system (OS), while the other executes a bare-metal application directly on the hardware without any operating system. A lightweight application program interface (API) framework is used to exchange information between the two cores thanks to the On-chip Shared Memory.

### Linux core

The Linux OS is loaded from the SD Card thanks to the relative interface available in the Processing System. Once started the OS is in charge of three main tasks, to housekeep the entire processing system, to guarantee a high-level interface with the external user and to provide network connectivity. The housekeeping duties involve:

- initialization of the entire system;
- configuration of the programmable logic clocks with the right frequencies;
- management of the bare-metal core, setting the software parameters and controlling the execution flow trough start and stop commands.

The hardware configuration of the programmable logic and the software running on the bare-metal core can be updated at any time through the high-level interface. This allows to perform a partial or full reconfiguration at any time, adapting the system to different experimental configurations. Moreover, through the OS it is possible to store significant data in the SD Card to be later used for further analysis. The network connectivity is achieved using an Ethernet port that can be connected either to a local area network (LAN) or directly to the Internet. The Ethernet connection allows to perform the same commands provided by the high-level interface remotely. This feature is of great use for debugging and testing and was largely exploited; indeed while the experimental setup was assembled at the Italian Institute of Technology in Genoa, I did most of the corrections and adjustments remotely from Cagliari. The network connectivity is also used to transfer the online results to the Offline Processing unit for comparison and validation.

#### Bare-metal core

The bare-metal application running on the second core is in charge of the final elaboration steps to decide the stimulation to send back to the neural network. It is important to note that the code for the closed-loop algorithms can be written with general-purpose programming languages (I used C), thus making it very easy to be changed and adjusted by the general users without a background in Electronics Engineering. This is a significant advantage of this system, making it versatile and reusable for different experiments. As already said before, the loop is closed based on the neural activity detected on the network and stored in the external DDR Memory, the access to the memory is guaranteed by a direct connection to the DDR Controller that allows to read the required data in real-time. The different closed-loop algorithms that have been implemented and tested will be presented in Chapter 6 relative to the experimental results. The only constraint that must be respected by any algorithm aiming at closing the loop is to be short enough to be executed in real-time in the bare-metal core. This means that each iteration of the algorithm must take a time equal or less than to the sampling time $T_s = 1/f_s$. With a sampling rate of 18 kHz this time is 55.56 µs which translates in around 37000 processing cycles of the processor clocked at 667 MHz.

### 3.4.3 DDR Memory

Besides the Zynq®-7020 APSoC the ZedBoard™ features 512 MByte of DDR3 memory connected with a 32-bit bus clocked at 533 MHz. The theoretical bandwidth is thus:

$$\begin{aligned} BW_{DDR3} &= f_{clock} \cdot 2 \cdot bus\_width \\ &= 533\,\mathrm{MHz} \cdot 2 \cdot 32\,\mathrm{bit} \approx 35\,\mathrm{Gbit/s} \end{aligned} \tag{3.10}$$

Where the factor 2 comes from the fact that the memory is double data rate. However, the practical memory bandwidth will be lower, limited by the DDR controller algorithms and the access patterns.

This external memory is shared among the three different processing entities in the system: the Linux core, the bare-metal core, and the Programmable Logic & Hardwired Blocks. To facilitate the sharing of this resource, the memory has been virtually split into four parts. The first partition, the bigger one, is dedicated to the operating system, and it is accessible only by the Linux core. The second partition instead is exclusive for the code of the bare-metal core. Another partition contains the filtered samples and the information on the detected spikes, and of course, it is written exclusively by the Threshold Updating & Spike Detection module, while it can also be read by the ARM® cores. This partition is treated as a circular buffer meaning that, when the end is reached, the module starts over at the beginning overriding the oldest samples and spikes' flags. The last slice of memory can be written and read by all the processing entities, and it is used to exchange information between them. In this part are, for example, stored the images to be projected by the HDMI Controller. The size of each memory partition depends on the experiments and algorithms that must be executed, and can be changed easily to adapt the system. I did not notice any significant impact on the overall system performances due to interferences among the different players sharing the same DDR memory.

### 3.4.4 SD Card

Another external memory featured in the Zyon is the SD Card, much slower than the DDR but also much bigger. In the current implementation, a 8 GByte Card is used, but smaller or bigger cards can be used as well. The SD Card is a non-volatile memory (differently from the DDR), and, for this reason, it is responsible for holding all the software and hardware configurations when the system is shut down, which include:

- the operating system to be executed on the Linux core and all the relative settings instructions;
- the closed-loop algorithms for the bare-metal core;
- the hardware bitstream containing the instruction to configure the Programmable Logic.

In addition, it can contain pre-generated pictures to be projected by the HDMI Controller, or it can be used to store the online results produced by the system.

### 3.4.5 HDMI Transmitter

Finally, the last component of the Zyon unit is an HDMI Transmitter that controls the HDMI interface. As the Image Creator and the HDMI Controller, also this component is used only when optical stimulation is needed. Specifically, the component used is the ADV7511 by Analog Devices, and, even if now it is set to project 608x684 pixels images, it can supports much higher formats, up to 1920x1080 pixels (commonly referred as 1080p or Full HD).

# Chapter 4

# Noise reduction

*"The amount of noise that anyone can bear undisturbed stands in inverse proportion to his mental capacity and may therefore be regarded as a pretty fair measure of it. Noise is a torture to all intellectual people."*
Arthur Schopenhauer (Philosopher)

## 4.1 3·Brain, almost 4

In this chapter, I will present the work done during my abroad period of six and a half months in the wonderful Switzerland, that I must admit is almost as good as Sardinia, but not as much. There I stayed in Wädenswil, near Zurich at the 3·Brain AG headquarters. As already said before this company produces the HDMEA-based acquisition systems used for the experimental setup created during this PhD. My task there was to develop digital processing algorithms to be implemented on an FPGA to improve the signal-to-noise ratio (SNR) of the acquired biological signals by exploiting oversampling techniques. The primary goal is to achieve noise reduction in the signals from the whole array in real-time.

The chapter is organized as follows: the section 4.2 summarizes the state-of-the-art of noise and denoising in MEA acquired signals. The next two sections describe in detail the specific problem challenged in this work. Sections 4.5 and 4.6 describe two possible approaches to solve the problem. Finally, section 4.7 deals with the feasibility of the selected approach in real-time, and section 4.8 with its generalization on all the HDMEA acquisition systems.

## 4.2 Noise, noise everywhere

The signal-to-noise ratio is of great importance for any acquisition system; it defines how much power of the acquired signal can be imputed to useful information with respect to the background noise always present.

> **Signal-to-noise ratio**
> (Oxford English Dictionary, 2018): (a) the ratio of the strength of a desired signal to that of unwanted noise or interference; (b) colloq. (chiefly Computing) a measure of how much useful information there is in a system, such as the Internet, as a proportion of the entire contents.

The SNR can be defined as:

$$SNR = \frac{P_{signal}}{P_{noise}} = \frac{A_{signal}^2}{A_{noise}^2} \tag{4.1}$$

where $P_{signal}$ is the power of the signal that carries useful information, in our case the biological signal of interest, while $P_{noise}$ is the noise power which conveys useless information. $A_{signal}$ and $A_{noise}$ instead represent the root mean square (RMS) amplitude of the signals. When acquiring from microelectrode arrays the input signals suffer from many different kinds of noise (Liu et al., 2015; Obien et al., 2015):

- Thermal noise (also known as Johnson–Nyquist noise): for temperatures above absolute zero (0 K or -273.15 °C) the electrons inside the electrodes, as in any electrical conductor, suffer from thermal agitation, thus producing an electrical noise. This noise is proportional to the electrical resistance of the electrodes, the frequency bandwidth in which is measured and, of course, the temperature (Liu et al., 2007).

- Biological interferences: any biological electrical activity present in the input signal that is not of interest in the current recording. This includes the activity of distant neurons or nearby muscles (only for *in vivo* experiments). Moreover, when considering action potentials then local field potentials are an interference and vice versa.

- Electrical interferences: any electrical artifact present in the input signal generated by the surrounding environment such as the 50-60 Hz interference due to power lines.

- Flicker noise (also known as 1/f noise): significant only at very low frequencies, it is caused by impurities in the conductive channels, and it is present in practically any electronic device.

- Non-linearity noise: all the noise contributions due to the limitations in the electric circuitry used to acquire the signals. The front-end amplifier is one of the main contributors for this noise; another is the quantization error introduced by the analog-to-digital converters, that is usually approximated as $\frac{LSB}{\sqrt{12}}$, with LSB being the least significant bit amplitude.

Reducing the input noise is one of the main challenges in producing good acquisition systems. Over the years, many different techniques have been developed in order to improve the SNR of the input signals. The usual target is to acquire a meaningful signal with an amplitude at least 5 times higher than the background noise (Obien et al., 2015).

The most significant way to improve the SNR is to reduce the resistor at the electrode level. It is also important that the input impedance throughout the entire array is constant to acquire good quality signals. After passing through the electrodes, the signals reach the front-end amplifiers. Sufficient area and power must be allocated to get the required low noise performance in the very first amplification stage. The power limitation is due mainly to heating problems caused by energy dissipation; the area instead is fixed by the electrode density. The available power and area per electrode are limited in HDMEA integrating thousands of sensor sites.

To overcome such limitations in the hardware front-end, a lot of digital techniques have been developed to manipulate the biological signals enhancing the interesting features and discarding the noise. First of all, usually the two main components of the extracellular activity, action potentials and local field potentials, are split and studied individually to avoid interferences between them. Since these two components occupy different ranges of the spectrum frequency, a band-pass filter is used for the purpose. The spikes occupy the spectrum between 300 Hz and 6 kHz; the LFPs, instead, are below 300 Hz (Gibson et al., 2012). The band-pass filter is also a very useful tool to get rid of all the noise outside the frequency band of interest, and it is practically always used. More advanced denoising techniques have been developed targeting specifically either of the two neural activities. For the action potentials, many algorithms to overcome the low signal-to-noise ratio are based on the instantaneous energy of the signals. Such class of algorithms includes the nonlinear energy operator (Kim and Kim, 2000) and the Teager energy operator (Choi et al., 2006). This kind of algorithms is widely used for their low computational requirements. Another set of algorithms, instead, are wavelet-based such as in (Nenadic and Burdick, 2005) and (Liu et al., 2015). Other techniques instead target the denoising of the LFP. In this field, most of the algorithms target the removal of artifacts present especially in *in vivo* experiments. Some algorithms are based on the stationary wavelet transform such as (Islam et al., 2014). Other methods are instead based on the concept of adaptive noise canceling (ANC) such as (Xinyu et al., 2017).

Differently, from the algorithms presented above, the denoising technique that will be presented in the following is not focused on improving the quality of a specific kind of neural activity, but rather on reducing the overall thermal noise in the acquired biological signal. The resulting data, with a higher SNR, can then be processed with other algorithms to achieve even higher quality.

## 4.3 Problem description

The aim of this approach is to reduce the thermal noise in the signal. As already anticipated the thermal noise is proportional to the electrical resistance of the electrodes, the absolute temperature, $T$, and finally the frequency bandwidth over which we are considering the noise, $\Delta f$. The exact equation to calculate the thermal noise is:

$$v_n = \sqrt{4 \cdot k \cdot T \cdot Re(Z_e) \cdot \Delta f} \tag{4.2}$$

where $k$ is the Boltzmann constant and $Re(Z_e)$ represents the electrical resistance, the real part of the total impedance $Z_e$, (Liu et al., 2007). The equation 4.2 holds true for any electrical conductor. Right after each electrode, the acquisition chip presents a front-end amplifier in order to boost the biological signal and reduce the interferences of noise sources. As long as the density of the electrodes is sufficiently small, the available area, to implement the front-end amplifiers, is sufficient to achieve a good SNR. However, as we transition from MEA to high-density MEAs, more problems start arising. With densities up to thousands of electrodes per square millimeter, the available area for the underlying electronics is not sufficient anymore to implement the full amplification-digitization path for each electrode while maintaining a good SNR (Tsai et al., 2017). A widely used solution to such limitation is to exploit time-division multiplexing of the signals and reuse the same electronics for multiple electrodes. Consequentely, more area per component is available. This technique is used both for *in vitro* recording systems as in (Berdondini et al., 2009; Tsai et al., 2017) and for *in vivo* as in (Angotzi et al., 2018; Lopez et al., 2014). The resulting acquisition system is something similar to the schematic of Fig. 4.1. Each electrode has its own front-end amplifier with a very limited area, for this reason a second-stage amplification is provided before the digital conversion. For the second-stage amplification, the signals from multiple electrodes are multiplexed in time, and a single amplifier with a much higher bandwidth is used. The

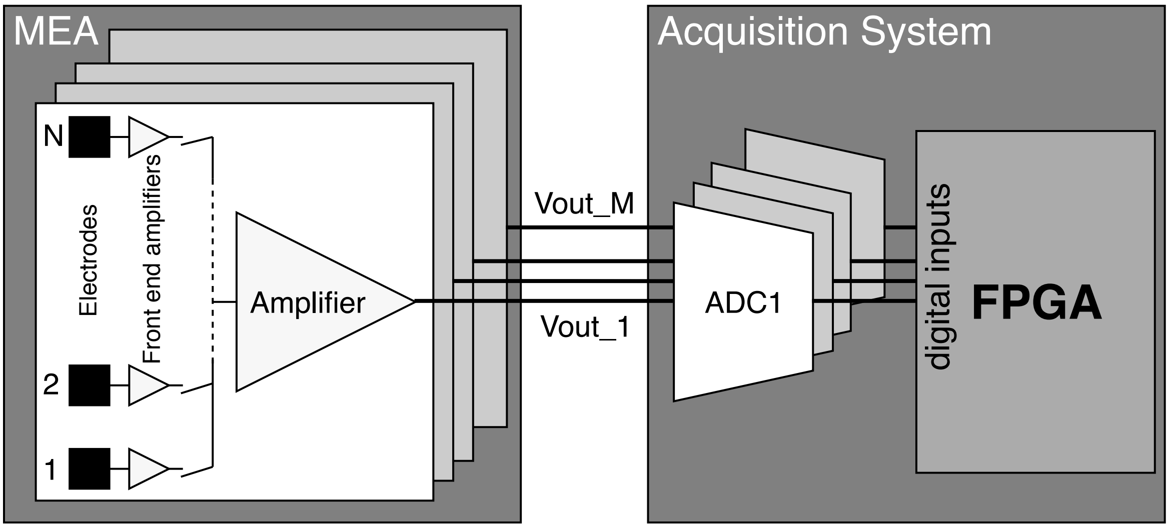


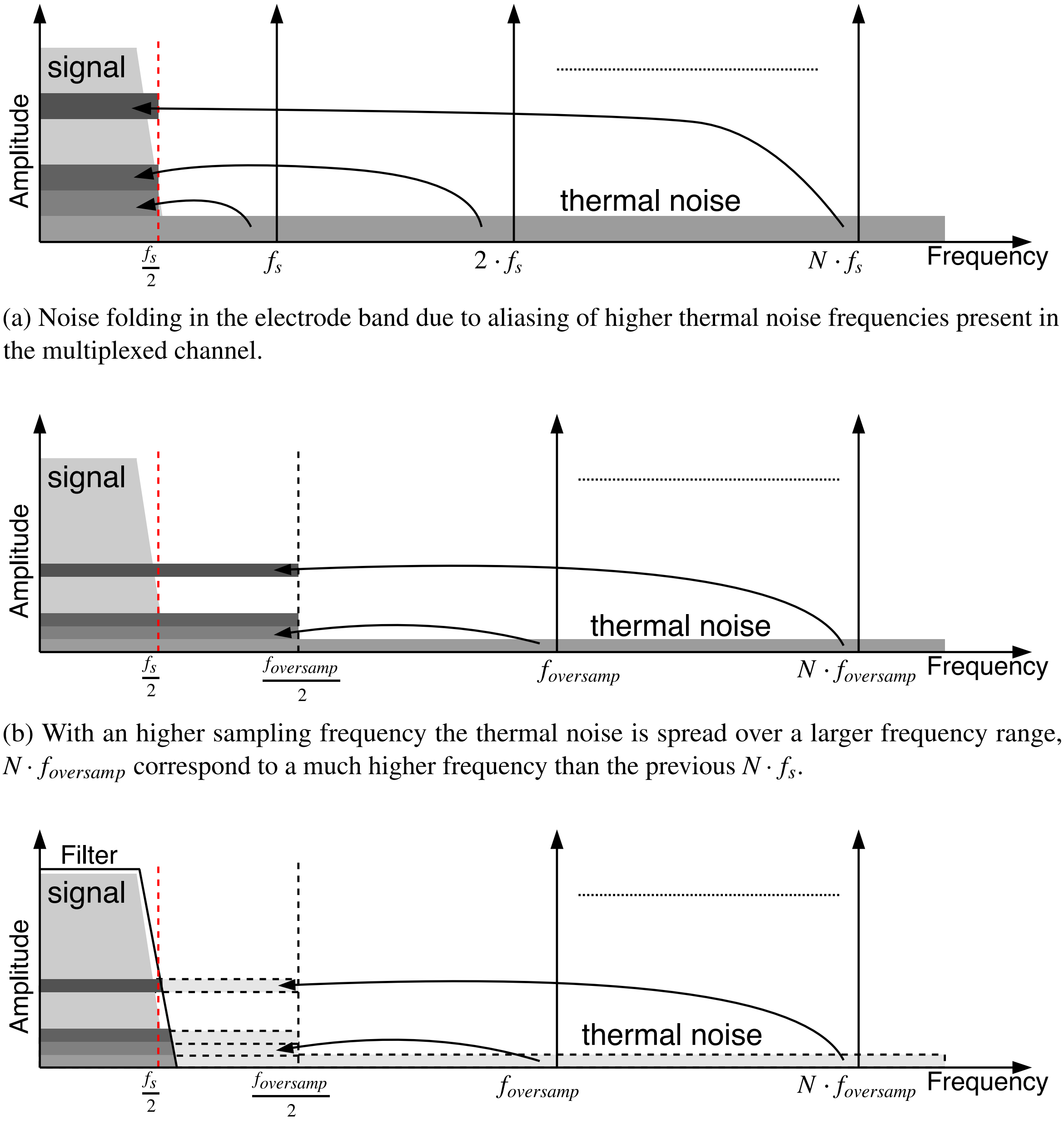
Figure 4.1 Schematic example of an HDMEA acquisition system with two stage amplification. The time-division multiplexing among multiple electrodes is exploited; each Vout carries multiple biological signals.

amplified output signal is then sampled by an analog-to-digital converter (ADC) and sent to a processing architecture (here an FPGA) that is in charge of demultiplexing the signals.

The major drawback of this approach is related to the thermal noise, in order to understand this easily, it is opportune to see the problem from the frequency domain. Let us consider a sampling frequency per electrode equal to $f_s$, then the required bandwidth for the amplifier and for the ADC is $N \cdot f_s$, because they have to carry $N$ time-division multiplexed signals. As a result, we have that the thermal noise spreads in a much larger $\Delta f$, indeed in the multiplexed channel, it occupies a bandwidth $N$-times that of the single electrode signal. Normally, everything outside the frequency band of interest can be filtered out in the digital domain, but not in this case. Indeed each electrode is sampled with just $f_s$. Thus the thermal noise is folded multiple times in the frequency band of interest and therefore becomes indistinguishable from the biological signal; a situation usually referred to as the aliasing effect. The graphical representation of what just said is reported in Fig. 4.2a. Adding some numbers on it can help to put things into perspective: $f_s$ is usually around 10-20 kHz, $N$ can reach values of few hundreds meaning that $N \cdot f_s$, the ADC sampling frequency, is in the order of MHz.

The approach adopted to reduce the impact of such folded noise is to increase the ADC sampling frequency thus the $f_s$ of each electrode passing to a higher $f_{oversamp}$. We define the oversampling ratio (OSR) as:

$$OSR = \frac{f_{oversamp}}{f_s} \tag{4.3}$$



(a) Noise folding in the electrode band due to aliasing of higher thermal noise frequencies present in the multiplexed channel.

(b) With an higher sampling frequency the thermal noise is spread over a larger frequency range, $N \cdot f_{oversamp}$ correspond to a much higher frequency than the previous $N \cdot f_s$.

(c) The noise outside the signal band of interest can be filtered out.

Figure 4.2 Noise folding in the electrode band due to aliasing of higher thermal noise frequencies present in the multiplexed channel.

In Fig. 4.2b, it can be seen that the noise is spread over a much broader frequency band. Indeed, $N \cdot f_{oversamp}$ is much higher than $N \cdot f_s$. Now that $f_{oversamp}$ is wider, not all the noise is aliased into the relevant signal band and can thus be filtered out in the digital domain, as depicted in Fig. 4.2c. Since the thermal noise, up to frequencies of $10^{12}$ Hz, has a flat power spectrum, it can be, with good approximation, considered as white noise (Barry et al., 2012). The most straightforward technique to filter out the white noise after oversampling is averaging, downsampling the signal back to the original frequency $f_s$ while removing the white noise. The maximum improvement in the SNR that can be achieved with this technique is calculated by comparing the non-filtered noise power with the filtered one. With reference to the Fig. 4.2b, the energy spectral density, $E_n^2(f)$, of the white noise folded into the $\frac{f_{oversamp}}{2}$ band, can be described with the equation:

$$E_n^2(f) = e_{rms}^2 \cdot \frac{2}{f_{oversamp}} \tag{4.4}$$

with $e_{rms}^2$ being the average noise power. It is easy to notice that $E_n^2(f)$ decreases as $f_{oversamp}$ increases. The non-filtered noise power is:

$$P_{noise} = \int_0^{\frac{f_{oversamp}}{2}} (E(f))^2 df = \frac{f_{oversamp}}{2} \cdot \left( e_{rms}^2 \cdot \frac{2}{f_{oversamp}} \right) = e_{rms}^2 \tag{4.5}$$

If we consider our biological signal to have the highest frequency component at $\frac{f_s}{2}$ and we filter everything outside of this band, then the noise power becomes:

$$P'_{noise} = \int_0^{\frac{f_s}{2}} (E(f))^2 df = \frac{f_s}{2} \cdot \left( e_{rms}^2 \cdot \frac{2}{f_{oversamp}} \right) = e_{rms}^2 \cdot \frac{f_s}{f_{oversamp}} = \frac{e_{rms}^2}{OSR} \tag{4.6}$$

Now, using the eq. 4.1 we can calculate the improvement in the signal-to-noise ratio as:

$$\frac{SNR'}{SNR} = \frac{\left( \frac{P_{signal}}{P'_{noise}} \right)}{\left( \frac{P_{signal}}{P_{noise}} \right)} = \frac{P_{noise}}{P'_{noise}} = OSR \tag{4.7}$$

So basically the OSR represents how much the SNR can be improved ideally, while the RMS amplitude of the noise is reduced by $\sqrt{OSR}$. More details about oversampling and averaging can be found in (Silicon Laboratories, 2013). Thus, one effective way to reduce the noise is to oversample the signal and average it subsequently.

## 4.4 «Would that it were so simple...»

It is not easy to perform a simple windowed averaging of the oversampled data in a time-multiplexed acquisition system. Looking back at the Fig. 4.1 it is easy to understand why, we are dealing with real electronics components, the amplifier does not hold an ideal response. Inherently, there is a transient period after the switching from one electrode to another, due to bandwidth-limitation of the shared amplifier. In the current commercial acquisition systems, the sampling of each electrode takes place only after the response has settled out, so after this transient period. If we increase the sampling rate of the ADCs, we obtain a situation as depicted in the Fig. 4.3. The blue signal is the time-multiplexed signal that goes from the amplifier to the ADC, with respect to Fig. 4.1, it is one of the Vout. The black arrows represent the switching between different electrodes. The signal was acquired with a GHz-oscilloscope connected to the 3·Brain BioCam X. In this system, the sampling frequency per electrode, $f_s$, is around 20 kHz. The amount of time per frame dedicated to each electrode is thus: $\frac{1}{f_s} \cdot \frac{1}{256}$ which is approximately 200 ns, as it can be seen in the picture. The transient response after switching occupies most of this time, with the signal reaching the new level only at the end of the time window where the signal is currently sampled, green dots. The red dots are instead obtained considering an oversampling frequency with $OSR = 16$. Almost all

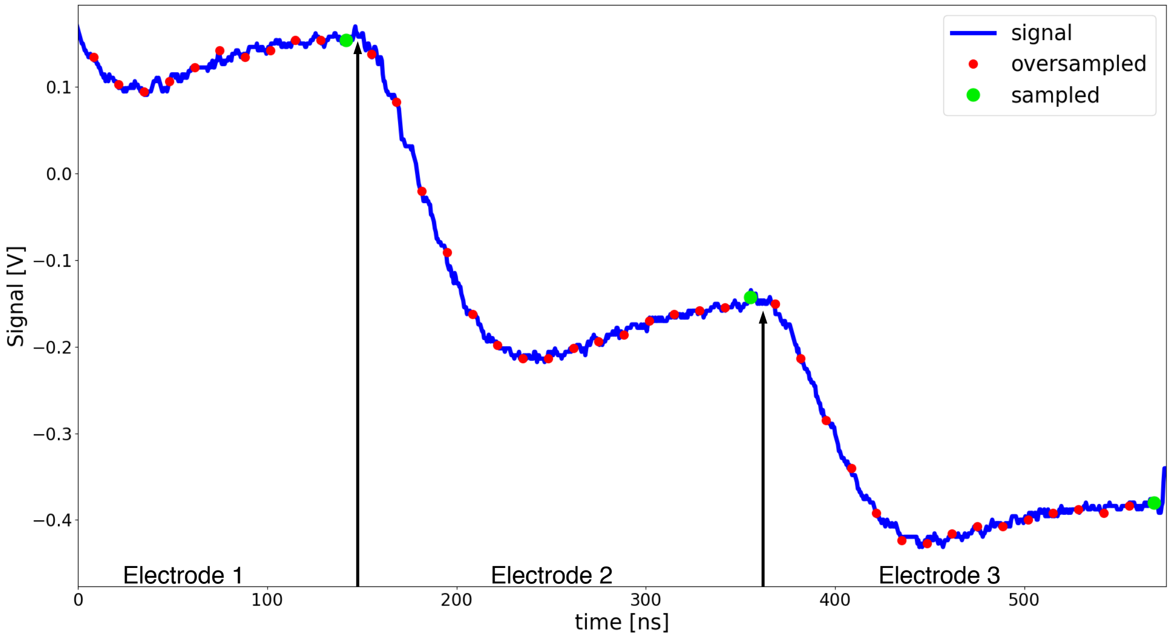


Figure 4.3 Transient response of the signal after switching between electrodes. The black arrows represent the instants of switching.

of these new samples are taken in the transient response and nothing good can be obtained by simply averaging them. So, what can we do? «It is . . . complicated.»

## 4.5 Least square fitting

SPOILER ALERT:
This approach did not work; you may want to skip to the next paragraph or enjoy my little fiasco described here.
SPOILER ALERT END
Basically, we have a bunch of data from the transient period, that we will call $y[i]$ with $i \in [1,\ 2,\ \dots\ ,\ OSR]$. We want to use this data in order to reduce the noise of the last sample $y[OSR]$. In order to achieve this, the adopted approach was to fit the oversampled data to a known model using a least squares algorithm. The fitting will produce the guessed model $y_{guess}[i]$. After fitting, the data can be normalized using the model so that all the samples are mapped to a constant level: $y_{normalized} = y[i] - y_{guess}[i] + y_{guess}[OSR]$. The normalized data can be averaged to reduce the noise as if there were no transient response from the bandwidth-limited amplifier. Briefly, the least squares algorithms aim at finding the parameters for the given model in order to minimize the sum of the squares of the residuals, i.e., the difference between the guessed model and the real data: $y_{residuals}[i] = y[i] - y_{guess}[i]$. Summarizing:

$$\text{least squares algorithms find } \ y_{guess}[i] \ \text{ so that: } \quad \sum_{i=1}^{OSR} y^2_{residuals}[i] \quad \text{is the minimum possible.}$$

Actually, after finding $y_{guess}$, the normalization and the averaging are completely useless for the denoising. Let us see why, by calculating $y_{denoised}$ as the average of the normalized samples:

$$y_{denoised} = \frac{1}{OSR} \cdot \sum_{i=1}^{OSR} y_{normalized}[i] = \frac{1}{OSR} \cdot \sum_{i=1}^{OSR} (y[i] - y_{guess}[i] + y_{guess}[OSR])$$

This can be written also like:

$$y_{denoised} = \left( \frac{1}{OSR} \cdot \sum_{i=1}^{OSR} y_{residuals}[i] \right) + y_{guess}[OSR]$$

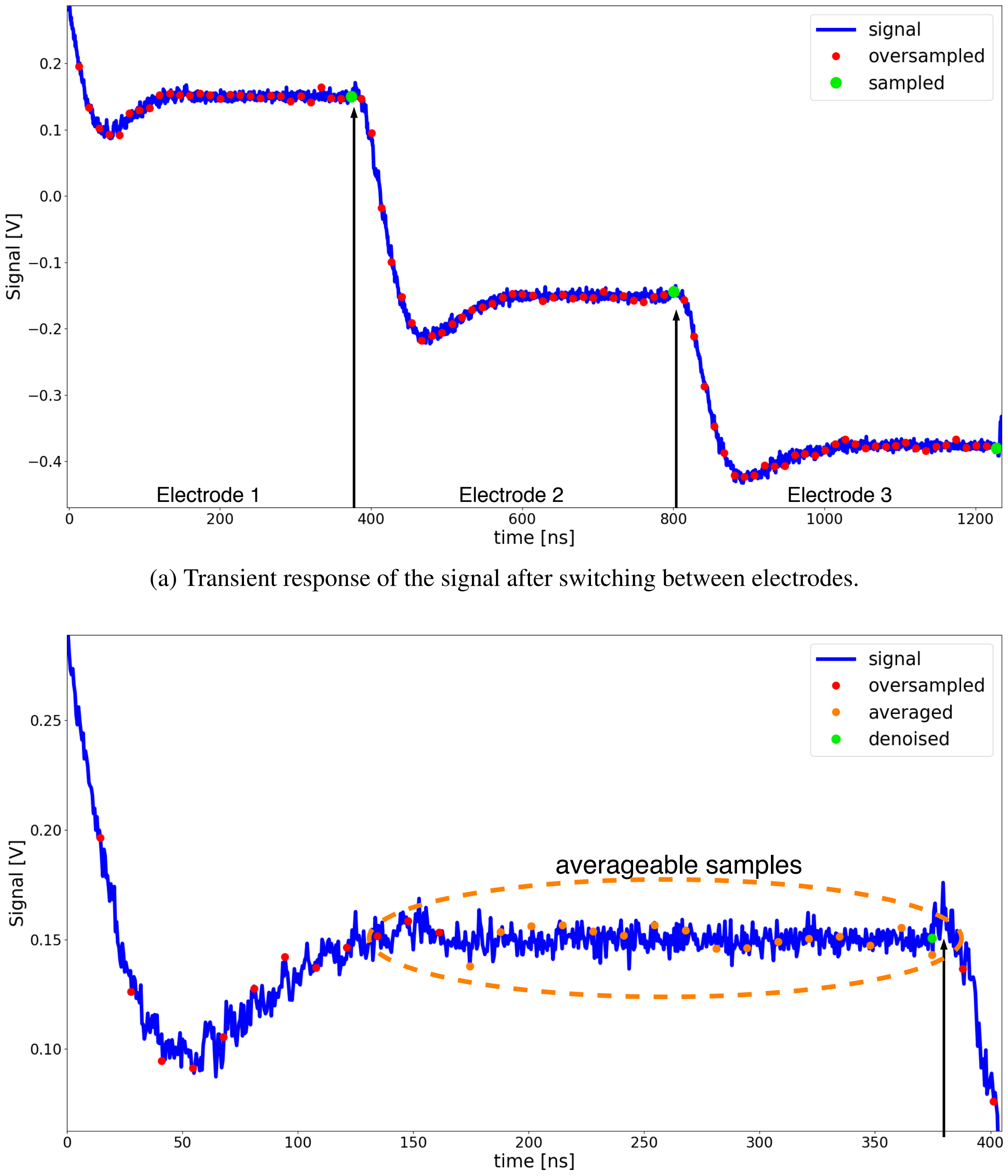
But the sum of the residuals is equal to 0, so basically:

$$y_{denoised} = y_{guess}[OSR]$$

A lot of offline experiments with real data have been done to evaluate the possible gain in the SNR using this approach. For this tests an $OSR = 16$ has been used; this is a plausible value for this kind of applications using ADCs running at under 100 MHz sampling frequency for around 250 electrodes multiplexed in time and originally sampled at 20 kHz. Different parameters, different models, different fitting algorithms have been tried, but nothing produced a satisfying improvement in the SNR. The average reduction in the noise level was around 1.3, instead of the 4 to be expected from the theory presented before. That least squares approaches are not so effective to increase the SNR was actually known since a long time (Enke and Nieman, 1976).
Let us go to plan B, actually more likely plan G, plans from C to F involving Kalman filters and other fancy stuff did not work as well and are not worth to be reported here.

## 4.6 SNR-frequency trade-off

The main idea is to reduce the switching frequency between electrodes. Thus after the transient period, we will have a much longer stable period before switching to the next electrode, and we can apply the oversampling and averaging approach. In Fig. 4.4a, we can see what happens in the 3·Brain BioCam X if we reduce the switching frequency between electrodes by half. Now, the switching time between electrodes is around 400 ns, with 256 electrodes multiplexed in times; this means that the total time for a frame is 100 µs resulting in a sampling frequency per electrode of 10 kHz. In return, maintaining the same sampling frequency for the ADCs, we have a lot of samples in the stable period that we can average to increase the SNR, as depicted in Fig. 4.4b.
The result is a trade-off between sampling frequency per electrode and input SNR. But can we really give up frequency to reduce the input noise? Most of the HDMEA-based acquisition systems currently commercially available present an acquisition frequency around 20-25 kHz per electrode, as we have seen in Chapter 2. However, such a high frequency is not always necessary. Indeed, for many analysis of neural circuits, a frequency of 10 kHz per electrodes is more than enough, and a higher frequency does not bring any significant improvement in the performance of the analysis (Navajas et al., 2014). So, for many applications, the $f_s$ per electrode of the acquisition system can be reduced by half without any significant reduction in



(a) Transient response of the signal after switching between electrodes.

(b) Averaged stable response.

Figure 4.4 Oversampling and averaging applied slowing down the switching frequency between electrodes.

the performance. Then, by applying oversampling and averaging, the SNR can be increased by a factor equal to the number of samples averaged, with the noise level reduced by the square root of the same value. The validity of this approach has been confirmed with offline experiments based on real data, in the next section we will see the feasibility of this approach online in an FPGA implementation.

## 4.7 Hardware implementation

It is possible to notice in Fig. 4.4b that not all the averageable samples have been used for the denoising. The explanation is actually quite simple. In order to average, we must sum up all the samples and then divide by their number. But divisions in hardware are not easy to implement unless the divisor is a power of 2, in those cases the division becomes a simple bit shift. For this reason, the hardware implementation can handle the denoising only if the number of samples to average is a power of 2; thus in the previous image only the last 16 acquired points have been used for the denoising.

The schematic of the final denoising system is depicted in Fig. 4.5. The design has been implemented on an Intel® Arria® 10 FPGA, model SX 270, which has an architecture similar to the Zynq APSoC seen in the previous chapter. The processing path to implement the averaging inside the FPGA is described in the following. The samples are acquired in the same order as depicted in Fig. 4.4a, first all the samples from the first electrode, then from the second and so on. A total of 256 electrodes are multiplexed in the same channel, and a total of 16 parallel channels are received by the FPGA. Each channel is sampled by a different ADC running at 80 MHz, so a total of 1.28 GSamples/s are received by the FPGA. With a $f_s$ around 10 kHz, 32 samples are acquired on each electrode before switching to the next. The Fig. 4.6 depicts the acquisition waveform from the FPGA point of view, with the clock running at 80 MHz. After acquisition, the last $n$ samples from each electrode are summed up in the Accumulator block, $n = 2^m$ is a power of 2 value that can be changed at run-time by the ARM processor. The result of each sum is then shifted to the right by $m$ bits in order to obtain the averaged samples. Note that, using a simple bit shift for a division, you completely lose all the decimal part of the result; an error that in the worst case is practically equal to the value of the least significant bit value. In order to mitigate this, I added a little rounding mechanism to the calculation of the average, in this way the maximum error is limited to half of the value of the LSB. This mechanism is implemented in the Averaging block of the figure.

The crucial point for the hardware implementation is to meet the real-time constraints. Once

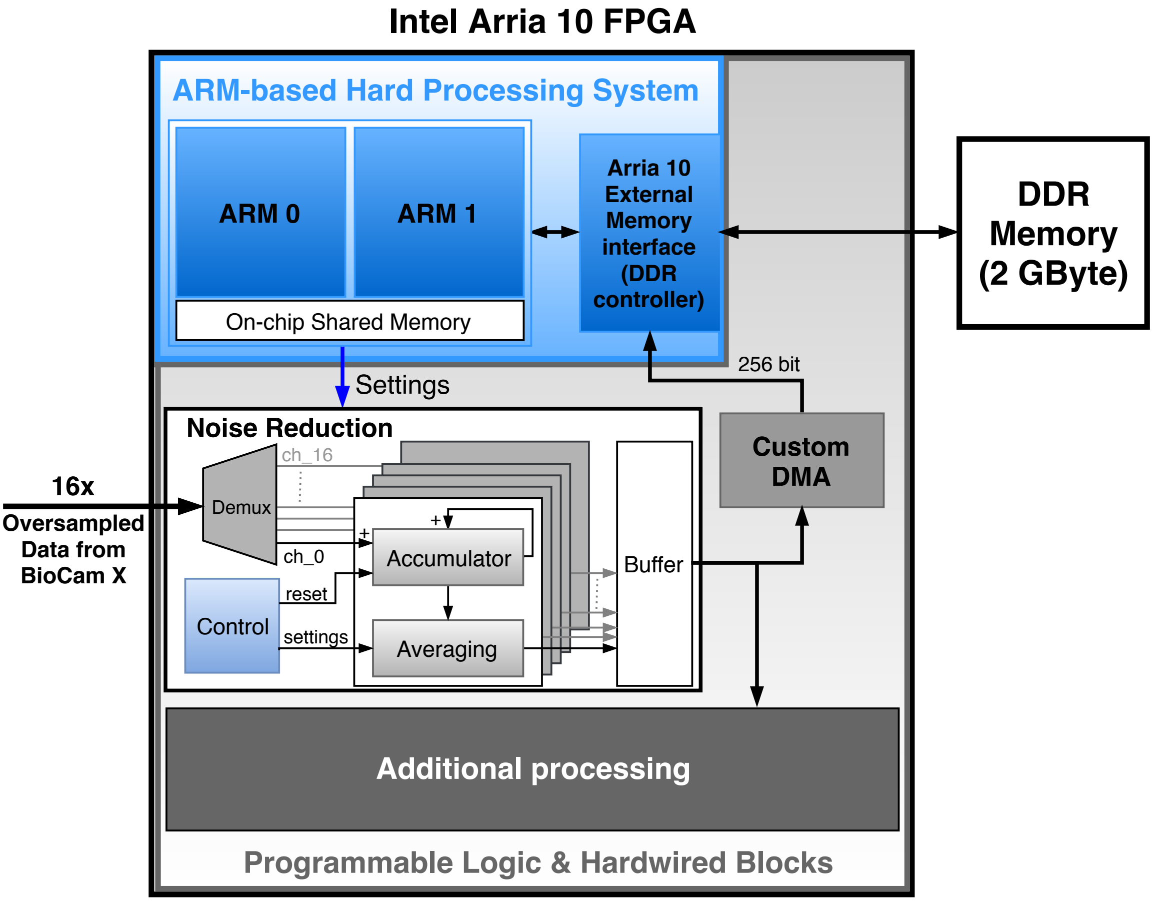


Figure 4.5 Hardware implementation of the denoising system.

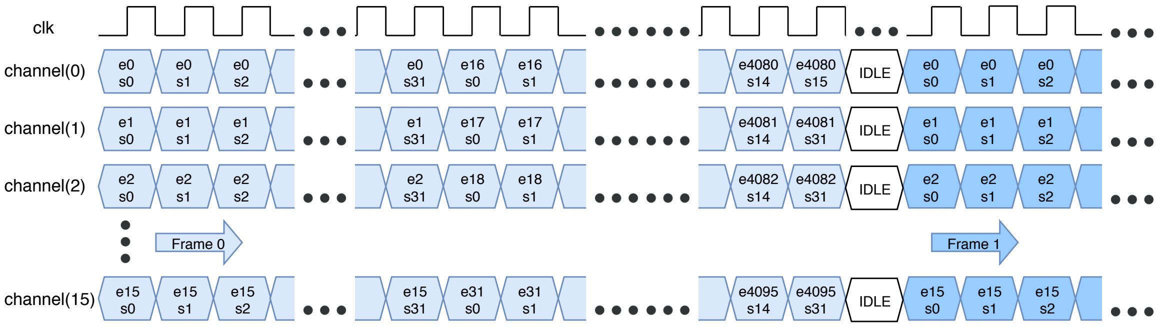


Figure 4.6 Oversampled input signals in the FPGA from the ADCs.

again, thanks to the internal structure of the FPGA, it is possible to implement multiple parallel paths to achieve the required throughput. As can be seen from Fig. 4.5, 16 parallel paths are used to perform the accumulation and averaging. The hardware implemented in the FPGA is clocked at around 80 MHz and each path is able to process one sample per clock cycle. Thus, with 16 paths, all the $1.28 \cdot 10^9$ Samples/s received from the ADCs can be processed in real-time.

After these steps, the denoised samples of the frame are temporary stored in a buffer with a size of 4096 samples. From here, the data takes two paths. One goes to a Custom DMA module, which is in charge of storing everything in the DDR Memory trough the Arria 10 External Memory Interface. Thanks to the 256 bit wide bus there are no bandwidth problems, and it is possible to store all the samples continuously. The other path, instead, goes to the Additional Processing part, where other blocks can perform further elaboration of the samples such as bandpass filtering. The final system has been successfully tested simulating the acquisition of real data, and it is currently under test in a real setup.

## 4.8 Generalization

The presented technique can be applied to all the HDMEA-based acquisition systems with a schematic similar to the one depicted in Fig. 4.1, without any particular analysis of the input signal and even without knowing the settling time of the amplifier. The general steps to follow to achieve a better SNR with this approach are the following:

1. Set the ADCs sampling frequency to the highest value allowed by the hardware components while still respecting the real-time constraints.

2. Set the desired switching frequency between electrodes, $f_s$, based on the application/-experiment requirements.

3. Calibrate the system by calculating the number of samples to average to achieve the best SNR. This can be done by fixing different constant signals in the different electrodes, and analyzing how the noise level changes averaging a different number of samples. The exact procedure is described in the Listing 4.1.

Listing 4.1 Calculating the number of samples to average to achieve the best SNR.

```
noise_rms_last = inf
for(m=0; m=m+1; m<=log2(OSR))
{
    n = 2^m //samples to average
    denoised_signal = averaging(signal, n)
    noise = denoised_signal - mean(denoised_signal)
    noise_rms = root_mean_square(noise)
    if(noise_rms < noise_rms_last)
        noise_rms_last = noise_rms
    else
        exit loop
}
n = 2^(m-1) //samples to average for best SNR
```

# Chapter 5

# Hardware results

*"It's important not to overstate the benefits of ideas.*
*I know it's kind of a romantic notion that you're just going to have this one brilliant idea and then everything is going to be great. But the fact is that coming up with an idea is the least important part of creating something great. It has to be the right idea and have good taste, but the execution and delivery are what's key."*
Sergey Brin (Co-founder of Google)

## 5.1 «Does it work?»

When I was still a master student, I had the luck to participate to a conference that taught me an important lesson. After finishing his speech, the guy in the stage asked the audience for questions, as it is usual. At this point, an old professor raised his hand and with a sardonic smile asked: "Very nice concept but... does it work? You didn't show any result!". The learned lesson was that the scientific community is mean and so I decided to apply for the PhD to be a part of it. Jokes aside, coming up with a good idea or concept is one thing, literally anybody can do it. Obtaining good results is a totally different matter, you need effort, competence, time, perseverance, sweat, blood, pain... ok maybe I am over-exaggerating now. I do not want to diminish the importance of ideas too much, but it is aggregating results that you obtain scientific progress, aggregating ideas you just get a minestrone. Moreover, communicating the results is perhaps as important as obtaining them in order to improve the overall human knowledge.

In the previous chapters, I have explained the main ideas behind this work, and how to realize them, now we will address the obtained performances and the road taken to obtain them. First, the application parameters were tuned to satisfy the required constraints in terms of accuracy of the algorithms and overall latency. Then I evaluated how the hardware parameters impact on the resources usage, and how they must be set to achieve the best fit in the selected platform, optimizing both performances and resources occupancy. But before starting the discussion, in Fig. 5.1, you can admire the working setup ("Yes, it works!").

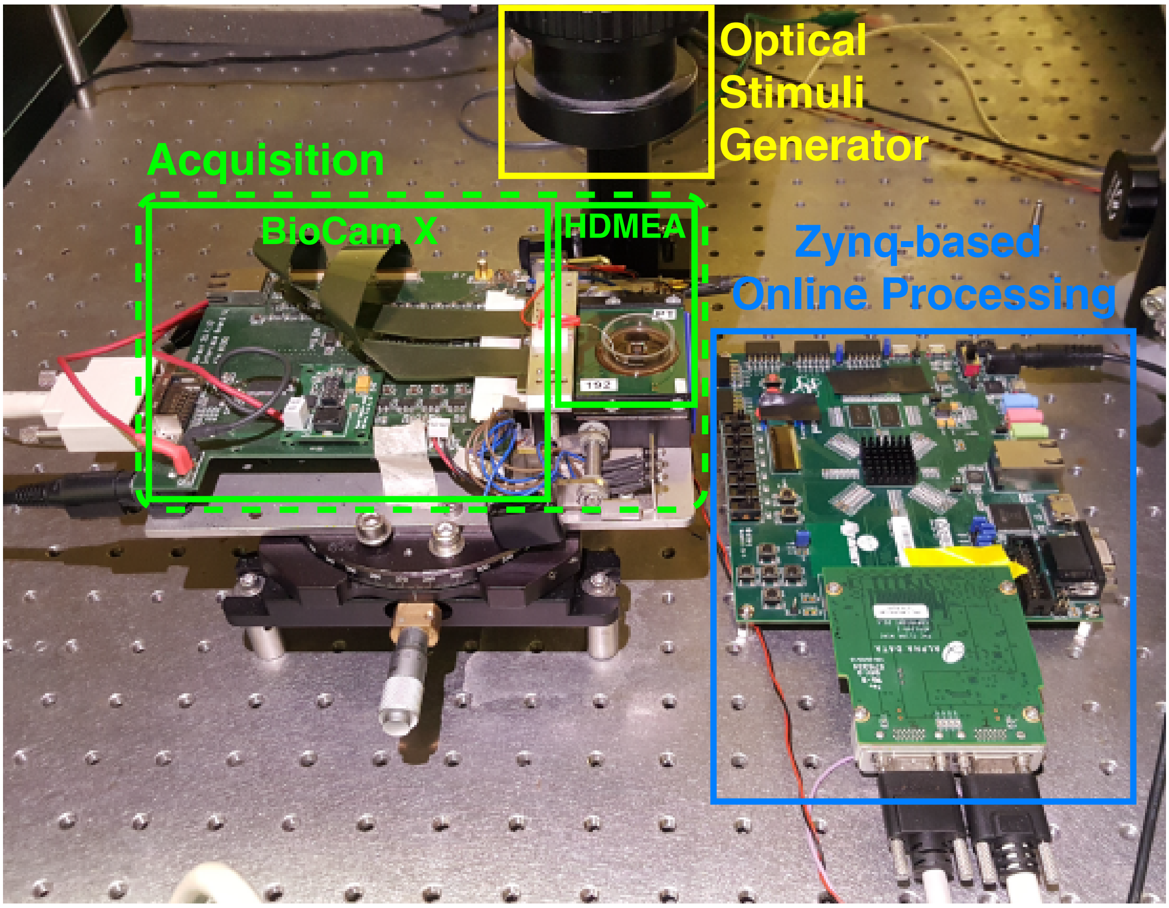


Figure 5.1 Experimental setup overview.

## 5.2 Design Space Exploration

When describing the processing tasks, in Chapter 3, it was evident that a lot of degrees of freedom where offered by the application, for example the order of the FIR filters or the threshold mechanism for the spike detection, but also by the hardware platform, for example how to distribute the processing tasks on the available resources. In order to exploit all the degrees of freedom, a detailed design space exploration (DSE) was performed. Three different experimental setup cases were considered, each targeting a different HDMEA acquisition platform to evaluate the different hardware needs for the Zynq-based Online Processing unit. Two of the acquisition platforms were actually really embedded in different versions of the experimental setup. One is the BioCam X, of which we talked extensively in the previous chapters and is the default acquisition system used. The other is a former version of the BioCam X, that was available at the Italian Institute of Technology. Compared to its newer counterpart, it presents the same number of channels, 4096, but a much slower acquisition rate of 7.8 kHz per electrode, thus less computational power will be required to process all the data, the full system is described in (Berdondini et al., 2009). Finally, the third one considered is the CMOS-MEA5000-System, which presents a slightly higher number of input channels and sampling frequency thus more computational power will be required. The main characteristics of the three systems used are summarized in Table 5.1. To simplify the discussion, they will be referred respectively as HDMEA1, HDMEA2 and HDMEA3, where a higher number denotes more computational power needed to process the data.

The main parameters that we are going to use to evaluate the performances are the spike detection accuracy and the total latency.

The accuracy depends on the algorithm used for the detection, but also on the frequency selectivity of the filtering stage, the better the filters, the higher the accuracy. Following the same notation used in (Navajas et al., 2014), we define the accuracy as:

$$Accuracy_{detection} = \left(1 - \frac{N_{misdetections}}{N_{spikes}}\right) \cdot \theta \left(1 - \frac{N_{misdetections}}{N_{spikes}}\right) \cdot 100\% \tag{5.1}$$

Here, $N_{misdetections}$ is the total number of mistakes in the detection, and it includes all the false positives (i.e. all the time the signal is classified as a spike but it is not), plus all the real spikes missed. $N_{spikes}$, instead, is the total number of spikes present in the signal. The

Table 5.1 Three different HDMEAs acquisition systems.

| Reference | Abbreviated name | Input channels | $f_{samp}$ [kHz] | Resolution [bit] |
|---|---|---|---|---|
| Berdondini et al. (2009) | HDMEA1 | 4096 | 7.8 | 12 |
| 3·Brain (2018) | HDMEA2 | 4096 | 18 | 12 |
| MCS (2018) | HDMEA3 | 4225 | 25 | 14 |

Heaviside step function, $\theta$, is used to set the lower limit for the accuracy to 0. It is easy to verify that, when there are no misdetections, $N_{misdetections} = 0$, the accuracy is 100%, while for $N_{misdetections} \geq N_{spikes}$ the accuracy is 0%. In order to use the Equation 5.1, it is mandatory to know exactly where the spikes are in the signal, so for the DSE, I used the simulated signals provided in (Quiroga et al., 2004). The signals were generated using 594 different spikes shapes extracted from real recordings. For the background noise, random spikes with random (smaller) amplitude were superimposed at random time in the signal, mimicking the real situation in an electrode. A total of four different data sets was used, each composed by four different tracks with a background noise amplitude respectively equal to 0.05, 0.1, 0.15 and 0.2 times the amplitude of the spikes. Among all the data sets, a total of 55330 spikes is present. Moreover, for each of the three different experimental setup cases, the signals were downsampled in frequency and scaled down in resolution to reflect the characteristics of Table 5.1. Additionally from the evaluations performed in this chapter with the simulated data, in the next Chapter 6, I will present an analysis with real data from retinal ganglion cells of mice.

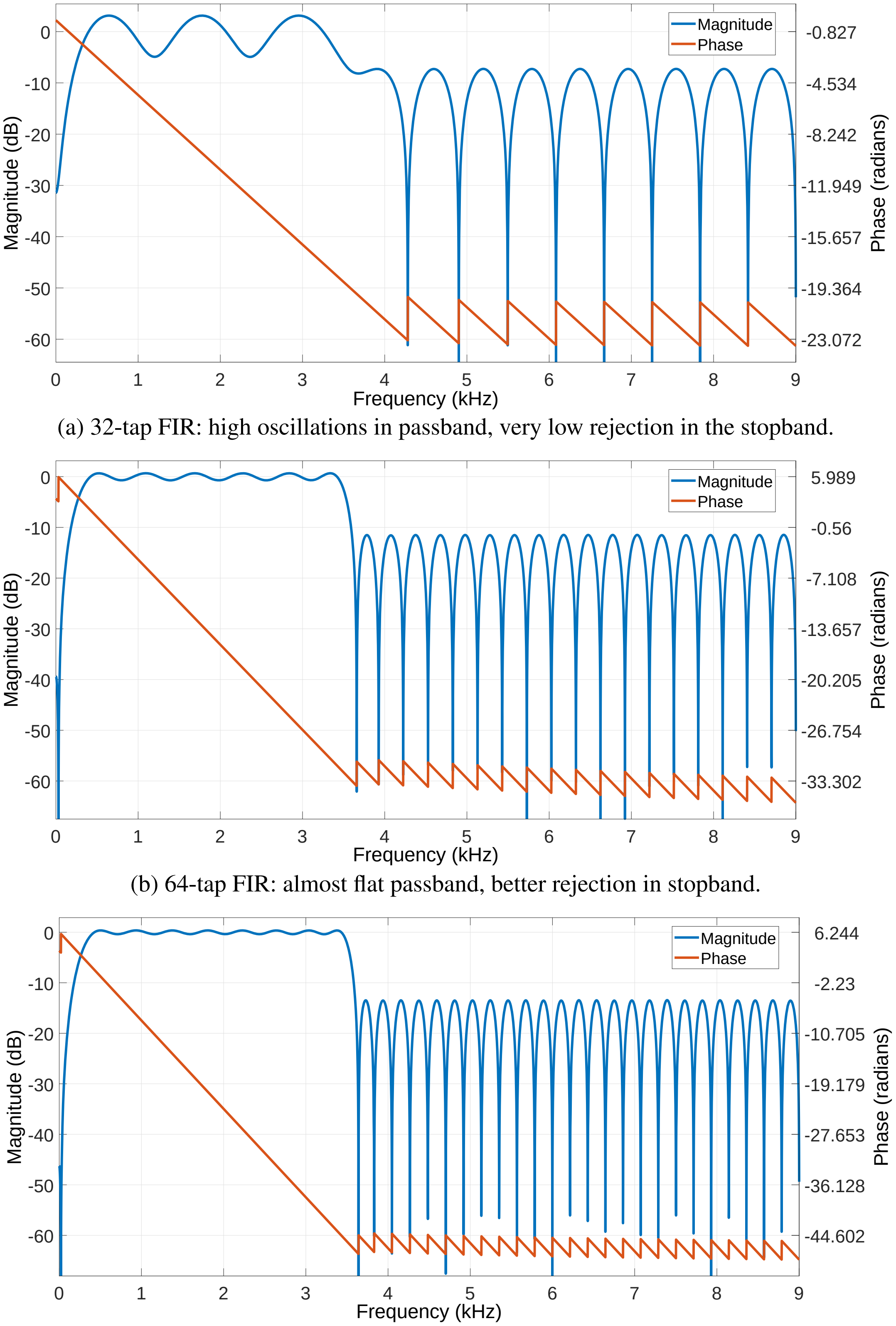
The latency, instead, is the sum of all the steps that must be done to reach a stimulation response. This involves not only the filtering and the spike detection but also the closed-loop algorithm and the various transfer time of the data.

## 5.3 Filtering Evaluation

The two main explorations done on the filtering stage regarded the filter order and the filtering partitioning.

### 5.3.1 FIR order vs detection accuracy

We already anticipated before that the FIR order is the most important parameter for the filtering stage. A higher FIR order means a better filtering response, which leads to better accuracy in the detection. This can be seen clearly in Fig. 5.2. Using only 32-taps (filter order 31) to design the FIR, Fig. 5.2a, we obtain a response with very high oscillations in the passband response. This means that the signal will suffer significant distortion during the filtering stage, that will also compromise the future processing. Indeed, any distortion in the signal will change the shape of the spikes, for example smoothing the peaks, which lead to increased difficulty in the detection. As we increase the order and we design our filter using 64-tap, Fig. 5.2b, or 80-tap, Fig. 5.2c, we can appreciate that the magnitude



(a) 32-tap FIR: high oscillations in passband, very low rejection in the stopband.

(b) 64-tap FIR: almost flat passband, better rejection in stopband.

(c) 80-tap FIR: flat passband, better rejection in stopband.

Figure 5.2 Magnitude and Phase response of FIR filters with different order.

Table 5.2 FIR order vs spike detection accuracy.

| Acquisition<br>System | $f_{samp}$<br>[kHz] | FIR taps<br>(Order + 1) | Latency<br>[ms] | Misdetections | Accuracy<br>[%] | Deviation<br>[%] |
|---|---|---|---|---|---|---|
| HDMEA1 | 7.8 | Ideal | - | 5369 | 90.30 | - |
| | | 64 | 4.10 | 5370 | 90.29 | 0.01 |
| | | 32 | 2.05 | 5389 | 90.26 | 0.04 |
| | | **16** | **1.03** | **5547** | **89.97** | **0.32** |
| HDMEA2 | 18 | Ideal | - | 3736 | 93.25 | - |
| | | 80 | 2.39 | 3736 | 93.25 | 0.00 |
| | | **64** | **1.78** | **3832** | **93.07** | **0.17** |
| | | 32 | 0.89 | 4119 | 92.56 | 0.69 |
| HDMEA3 | 25 | Ideal | - | 3564 | 93.56 | - |
| | | 72 | 1.44 | 3680 | 93.35 | 0.21 |
| | | **62** | **1.24** | **3706** | **93.30** | **0.26** |
| | | 50 | 1.00 | 3915 | 92.92 | 0.63 |

response in the passband becomes flatter and flatter. Moreover, the response in the stopband can be lowered, offering a better rejection of the unwanted frequencies (background noise). Another important feature that can be noticed in all the plots is the linear phase, that is also of mandatory importance to avoid distortions. The advantages of having a high order come with the cost of a higher processing power required and latency in the results. For the three experimental cases, I designed FIR filters (systolic multiply and accumulate architecture) with different orders and evaluated their performances in terms of accuracy of the subsequent detection and latency. A comparison with an (almost) ideal filter was used to empirically select the filter to implement. The ideal filter features a perfect response, flat passband and very high rejection in the stopband. It requires an infinite order to be implemented as an FIR; thus it is practically unfeasible, and it is used only to evaluate the drop in accuracy caused by the non-ideality of the order-limited filters. The final results are reported in Table 5.2, the criterion used to select the filter to be implemented on hardware is the following: the lower order which grants maximum 0.5% accuracy loss, compared to the ideal case.

The bold lines in the table represent the selected FIR filters. All of them feature a latency around 1 ms which is in accordance to the requirements for the closed-loop. Why those exact FIR order were selected for comparison will be explained in the next chapter.

### 5.3.2 Channel-to-FIR assignment exploration

We have seen in Chapter 3 that multiple parallel blocks are used for the filtering stage, each handling the processing of many channels in a time division multiplexing fashion. The resources occupancy of each block depends on the computational requirements, which are given by:

- sampling frequency;
- bit resolution of each sample;
- working frequency of the processing platform;
- FIR order;
- number of channels to process.

The first three parameters are fixed by the acquisition system and the processing hardware; the fourth was handled in the previous chapter; we have now to deal with the last parameter. The number of channels to process in each block must be chosen by the designer, and it will determine the number of blocks needed to process all the channels coming from the acquisition platform. It is easy to understand that, while each configuration leads to the same processing, they all have a different impact on the resource usage. As a practical demonstration, let us consider the HDMEA2 case with 4096 channels, sampling frequency of 18 kHz and the selected FIR order of 64-tap. Using the configuration where each FIR block process a single channel, the number of DSPs needed for the whole filtering explodes. At least one per module must be provided to perform the multiplications, thus 4096 DSPs in total, a number that is not featured even in the most expensive APSoCs. This is an example of very inefficient partitioning; in fact, each DSP is able to perform 83 Million MAC operations per second (one per cycle, clock at 83 MHz), but is in reality executing only $18\,\text{kHz} \cdot 32\,\text{MAC} \approx 0.6\,\text{Million}$ MAC per second. Providing each block with 2 channels to process, will increase the usage of the DSPs, while halving the block needed and thus the total DSP count. The purpose here is to find the partitioning which optimizes the resources usage. The last thing to say, before looking at the results, is that for a given partitioning, different FIR order can lead to the same resources occupancy. In those cases, it may be ideal to select the higher order which leads to better performances, and this is the reason for the specific values selected for comparison in the previous chapter. In Table 5.3 is reported the exploration done for the HDMEA2 experimental setup, for a 64-tap FIR, the best partition result in having 32 filtering modules processing 128 channels each. The same exploration

Table 5.3 FIR configuration vs resources usage (HDMEA2 use case).

| FIR taps | Channels | #FIR | Hardware resources | | |
|---|---|---|---|---|---|
| (Order + 1) | per FIR | needed | LUT | BRAM | DSP |
| up to 32 | 1024 | 4 | 12660 | 48 | 20 |
| | 512 | 8 | 6760 | 48 | 24 |
| | **256** | **16** | **4144** | **48** | **16** |
| | 128 | 32 | 4960 | 48 | 32 |
| | 64 | 64 | 6528 | 64 | 64 |
| up to 64 | 1024 | 4 | 25008 | 96 | 36 |
| | 512 | 8 | 13000 | 96 | 40 |
| | 256 | 16 | 7296 | 96 | 48 |
| | **128** | **32** | **4960** | **96** | **32** |
| | 64 | 64 | 6784 | 96 | 64 |
| up to 80 | 1024 | 4 | 31152 | 120 | 44 |
| | **512** | **8** | **16080** | **120** | **48** |
| | 256 | 16 | 10448 | 144 | 64 |
| | 128 | 32 | 8160 | 192 | 96 |
| | 64 | 64 | 7360 | 192 | 64 |

was done also for the other 2 experimental setup use cases. For the HDMEA1, with a 16-tap filter, the best partitioning is to have 1024 channels in each block, while for the HDMEA3, 62-tap, it is best to have 65 modules, each with 65 channels. The final hardware resources usage obtained with such partitioning for the three different use cases is reported in Table 5.6, and will be discussed later in this chapter.

## 5.4 Spike Detection Evaluation

Now that the parameters for the filtering stage have been optimized, we have to do the same for the spike detection stage, and then we can finally evaluate the final results.

Table 5.4 Window size vs spike detection accuracy.

| Acquisition System | $f_{samp}$ [kHz] | Window size [samples] | Window size [ms] | Misdetections | Accuracy [%] |
|---|---|---|---|---|---|
| HDMEA1 | 7.8 | 512 | 66 | 5642 | 89.80 |
| | | **1024** | **131** | **5293** | **90.43** |
| | | 2048 | 263 | 5369 | 90.30 |
| HDMEA2 | 18 | 2048 | 114 | 3736 | 93.25 |
| | | **4096** | **228** | **3712** | **93.29** |
| | | 8192 | 455 | 4072 | 92.64 |
| HDMEA3 | 25 | 2048 | 82 | 3838 | 93.06 |
| | | **4096** | **164** | **3564** | **93.56** |
| | | 8192 | 328 | 3735 | 93.25 |

### 5.4.1 Sliding window size exploration

A major parameter that can affect the performance of the spike detection algorithm implemented is the sliding window size, i.e., how many samples to use to calculate the standard deviation for the threshold. Of course, the optimal size is a measure of time, thus for the different use cases, the number of samples to use will vary depending on the sampling frequency. Indeed, the best accuracy was achieved for a window of 1024 samples for the HDMEA1 use case, with a sampling frequency of 7.8 kHz, and 4096 samples for the other two, which hold a similar sampling frequency of 18 and 25 kHz respectively. It is worth to remember that the window size can only be a power of two samples, due to hardware optimizations explained in Chapter 3. The full results of this exploration are reported in Table 5.4, where the optimal values are highlighted with bold text. From these findings it can be extrapolated that the optimal window size for the implemented algorithm is around 100-200 ms.

### 5.4.2 Dynamic threshold evaluation

All the explorations presented before were done with a rudimentary version of the spike detection algorithm featuring the dynamic threshold. Here instead, the same simulated datasets will be used to evaluate the accuracy performances of the final version of the algorithm with all the additional features, as described in Section 3.4.1. To demonstrate the advantages of having a dynamic threshold, a comparison with multiple static thresholds has been performed. The signal used were first filtered with an optimal filter in order to limit the variation in the accuracy to the detection method only. The results, depicted in Fig 5.3, will be discussed in the following. The first vertical bar in the chart represents the final version of the dynamic threshold implemented in the Zynq-based Online Processing unit of this work. The second bar is instead the best static approach possible. A static threshold was optimized offline for each different noise level of the signals. This approach, which is practically unfeasible, performed better than the dynamic threshold in some of the signals, but, overall, the total errors were still more. The following four bars represent a more realistic situation, the threshold is optimized offline but only for a particular noise level and then used to process all the signals. This would happen, for example, in the case of performing a tuning on the threshold level only at the beginning of the experiment. Any subsequent variation in the noise level due to the behavior of the cells, the degradation of the hardware or some external phenomena cannot be addressed by the static threshold. In all these cases, the obtained performances are significantly worse than the dynamic threshold, especially when the optimization was done for a low level of noise. The last bar on the chart is instead obtained with the algorithm described in (Quiroga et al., 2004). Being a detection performed without any optimization this lead to the worst results.

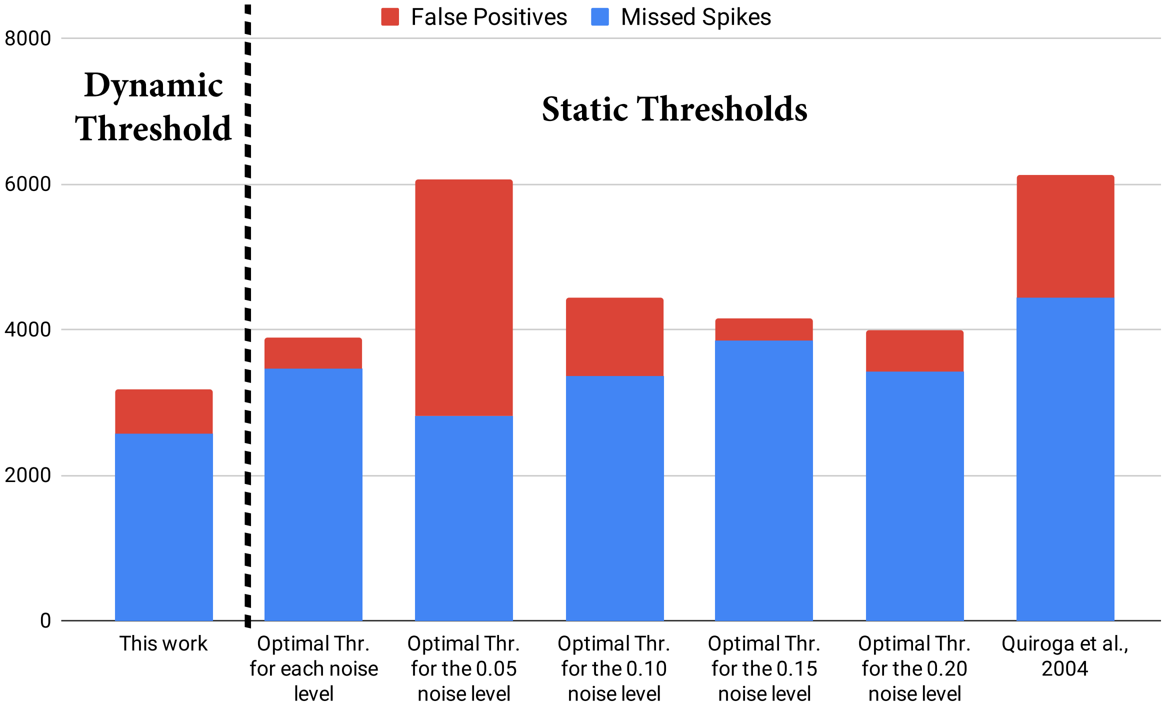


Figure 5.3 Spike Detection comparison.

Demonstrated the superiority of the dynamic threshold in ideal conditions, the last step is to evaluate the final accuracy of the implemented hardware system, with the 64-tap FIR filter. Using again the simulated signal with 55330 total spikes, the results are only slightly worse than what presented in the previous figure. The missed spikes are 2566 while the false positives 721, amounting to a total of 3287 misdetections. Using the Equation 5.1 we obtain an overall accuracy of 94.1%. Additional validation of the spike detection, performed on real data from a mouse retina, will be discussed in the next chapter that will be just about *ex vivo* experiments.

### 5.4.3 Spike Detection resources usage

In the description of the Spike Detection block in Chapter 3, it was reported that the calculation of the dynamic threshold involves 4 MAC operations for each new sample received. In the APSoC hardware, the MAC operations are usually implemented exploiting the DSP slices, that can perform one such operation per clock cycle. A clarification must be made now in order to later understand the resources usage of the Spike Detection block. In the Zynq family, the DSPs embedded are the 7 Series DSP48E1, which can perform multiplication with 25-bit x 18-bit data. Any time that a multiplication must be performed with larger data, more than one DSP is used. This is the case of the Spike Detection block, where the size of the data to multiply depends on the sample resolution, but also on the window size. Indeed, a wider window means that the sums needed to calculate the threshold will be larger, thus more bits will be multiplied later on. Moreover, in order to meet the real-time constraint without relying too much on buffers, 4 parallel paths have been designed, optimally using the 64 bit bus connection with the DDR Memory (4 samples of 16-bit each transferred each clock cycle). So, minimum 16 DSPs are necessary for a Spike Detection block, but as the sample resolution and the window size growth, more and more are needed. Finally, the total resources for the filtering and the detection stages for the different use cases are summarized in Table 5.5.

We can now compare the number of DSPs for the HDMEA2 use case with the workload figures for the filtering and the detection calculated in Equations 3.2 and 3.7. For the filtering, we have 32 DSPs that can perform a total of $32\,\mathrm{MAC}\cdot 83\,\mathrm{MHz} = 2.624\cdot 10^9\mathrm{MAC/s}$. Since $2.359\cdot 10^9\mathrm{MAC/s}$ are actually needed, they are used at 90% of their maximum performance. Doing the same for the Spike Detection, we have $28\,\mathrm{MAC}\cdot 83\,\mathrm{MHz} = 2.324\cdot 10^9\mathrm{MAC/s}$ available, while performing "only" $0.295\cdot 10^9\mathrm{MAC/s}$, resulting in a theoretical utilization factor around 13%. This result is due to two main factors. One is what I just explained above,

Table 5.5 Filtering and Spike Detection resources usage for the different use cases.

| Acquisition System | Module or Device | Hardware resources | | |
|---|---|---|---|---|
| | | LUT | BRAM | DSP |
| HDMEA1 | FIR (1024 ch/each) 4x | 858 | 6 | 1 |
| | | 3432 | 24 | 4 |
| | Spike Detection | 14475 | 14 | 24 |
| | Total | 17907 | 38 | 28 |
| | Target Device: Z-7015 | 46200 | 95 | 160 |
| | Usage | 39% | 40% | 18% |
| HDMEA2 | FIR (128 ch/each) 32x | 155 | 3 | 1 |
| | | 4960 | 96 | 32 |
| | Spike Detection | 15191 | 14 | 28 |
| | Total | 20151 | 110 | 60 |
| | Target Device: Z-7020 | 53200 | 140 | 220 |
| | Usage | 38% | 79% | 27% |
| HDMEA3 | FIR (65 ch/each) 65x | 141 | 2 | 1 |
| | | 9165 | 130 | 65 |
| | Spike Detection | 15059 | 14 | 32 |
| | Total | 24224 | 144 | 97 |
| | Target Device: Z-7030 | 78600 | 265 | 400 |
| | Usage | 31% | 54% | 24% |

that more than one DSP is used to perform some multiplications that do not fit the 25x18 multiplier featured in the DSP48E1. The second factor is that the DSPs resulted from the beginning the less used resources, so more effort was put to minimize buffering, control, and routing, that consume LUTs and BRAMs. This was also done at the cost of "wasting" DSP cycles, thus the low utilization factor. The table also reports the smaller chip of the Zynq®-7000 SoC Family required to fit the design, respectively the Z-7015, the Z-7020 (embedded in the Zedboard™), and the Z-7030. The three chips are low-range APSoC commercially available with limited cost. All the chips feature a dual-core ARM® Cortex™-A9 processor and a 28 nm Xilinx programmable logic, with different availability of look-up tables (LUTs) and hardwired blocks (BRAMs and DSPs). The percentage of resources occupied in each device is also reported in the table, with the BRAMs being the most used resources and the LUTs the second most used, confirming what previously said.

## 5.5 Hardware utilization

In conclusion to the resource usage analysis, in this section, I will present the final occupancy resulting from the developed working prototype. The full design implemented is the one previously described in Chapter 3 and depicted in Fig. 3.2. Besides the FIR filters and the Spike Detection module, it includes all the additional blocks for the correct working of the setup, such as the I/O Data Interface and and the TTL Generator, plus the modules required for the optical stimulation, the Image Creator and the HDMI Controller, and ,of course, all the interconnection buses. The final figures reported in the Vivado Design Suite, used for the implementation, are summarized in Table 5.6. The .5 in the BRAM column is due to the fact that each BRAM is actually composed by two smaller blocks that can be eventually used separately. As can be seen, all the additional modules amount only for a small percentage of the total resource usage. This means that for each use case, the remaining resources from Table 5.5 are enough to implement the full system in the target device. Moreover, the free resources still available can be used to improve the performances of the processing or to add additional features.

One last comment regarding the availability of DSPs must be made. Even if not practically tested, it is possible to perform the same processing with a much higher input sampling frequency without changing the design. Provided that the FIR order remains the same, this would be possible at the only cost of DSP resources without increasing the need for BRAMs and LUTs, that are the resources more close to saturation in the current device. With a little

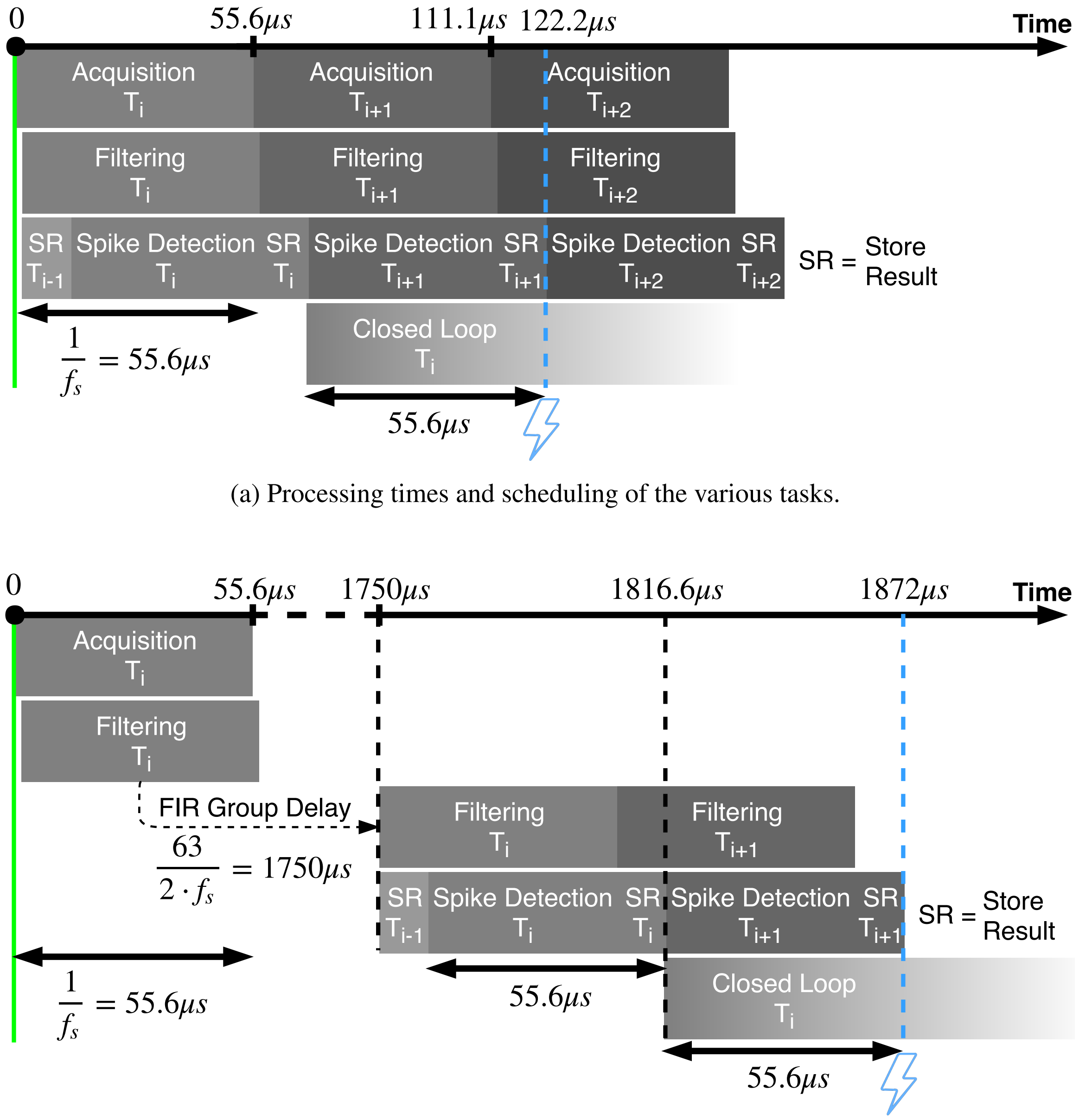
Table 5.6 Total resources usage for the developed prototype.

| HDMEA2 | Hardware resources | | |
|---|---|---|---|
| | LUT | BRAM | DSP |
| FIR + SD | 20151 | 110 | 60 |
| Other | 10745 | 12.5 | 5 |
| Total | 30896 | 122.5 | 65 |
| Z-7020 | 53200 | 140 | 220 |
| Usage | 58.1% | 87.5% | 29.5% |

analysis, I found out that a sampling frequency up to 30 kHz per electrode can be supported before saturating the bandwidth to the DDR memory offered by the AXI HP port (5 Gbit/s). A higher frequency would provide benefits in terms of accuracy, allowing to better filter out the unwanted noise, but also in terms of latency. In fact, as we will see in the next section, the major delay contribution is due to the filtering phase, and it is inversely proportional to the input sampling frequency.

## 5.6 Latency Evaluation

Since the system is conceived to operate in real-time in a closed-loop context, the elaboration responsiveness is as important as the correctness of the result. The stimuli generated as a response to a specific neural state must reach the biological cells while they are still in that state, in order for the closed-loop approach to make sense. In order to meet the real-time constraint while minimizing the latency, the elaboration system has been implemented as a pipeline synchronized with the input sampling frequency, $f_s = 18\,\text{kHz}$. The time period for each elaboration step is thus $\frac{1}{f_s} = 55.6\,\mu\text{s}$. In this time, an entire frame of 4096 samples, one from each input channel, must be processed by each task. The processing times were first evaluated from HDL simulations and then measured by means of a counter featured in the ZYNQ Processing System. The counter enables to measure time as number of clock cycles elapsed from a given trigger, being clocked at 333 MHz (half the ARM processors frequency), it gives a precision close to the ns. The resulting diagram of the scheduling is depicted in Fig. 5.4a. The processing begins with the acquisition of the samples performed by the I/O Data Interface, an entire frame is received from the HDMEA platform in 55.6 µs. Without waiting for the entire frame, as soon as the first samples are received, they are passed to the filtering stage; the introduced delay of this module is negligible, amounting to only few clock cycles of the programmable logic clocked at 83 MHz, we are in the order of ns. Thanks to the exploited parallelism, the filtering stage is completed slightly after the reception of the last sample of a frame. The delay introduced is once again negligible. For the Spike Detection, instead, we have an additional delay that must be considered, due to the sharing of the same bus to both store the results and read the samples required to update the threshold. The spike detection can only begin after around 11 µs from the completion of the previous detection step, this time is needed to store the previous results to the DDR memory. Anyway, the module is so fast processing that must stall waiting for the production of filtered samples to finish. The detected spikes are available to the closed-loop algorithm right after the storage part just mentioned. As represented in the picture with the extremely cool fading effect, the



(a) Processing times and scheduling of the various tasks.

(b) Final latency considering also the group delay of the FIR filter.

Figure 5.4 Scheduling and latency for the final prototype of the system with an input sampling frequency of 18 kHz and FIR order of 63.

closed-loop processing time is not strictly fixed but depends on the used algorithm. The ideal situation would be to have a closed-loop processing time equal or less than the frame period of 55.6 µs, in this way the real-time constraint is always met. In the picture, this upper bound limit is represented by the dashed blue line. However, it can happen to have a closed-loop algorithm with extremely variable latency. In such case, a worst-case scenario may lead to a violation of the real-time constraint in the immediate, but satisfaction of the constraint may still be met in the long run. This happened in one of the experiments that will be described in the next chapter.

Now that the scheduling of the processing has been deeply explained, it is time to introduce in the discussion the group delay of the FIR filters. Independently from the processing time, all the samples passing through the filtering stage carry an intrinsic delay due to the FIR response. This delay is calculated with the Equation 3.3, and amounts to 1750 µs; if needed it can be reduced by decreasing the FIR order at the cost of detection accuracy. Introducing this delay to the previous scheduling diagram, we obtain the new Fig. 5.4b with the final processing latency amounting to 1872 µs.

If the experiment involves electrical stimulation, then, after the closed-loop decision, a TTL trigger is instantaneously sent to the PlexStim™ Electrical Stimulator System. The delay introduced for the communication is negligible, while the time required for the electrical stimulus to be sent is 1 µs. Thus the total latency, from the signal acquisition to the stimulus reaching back the cells, is inferior to 2 ms.

Different steps must be performed when optical stimulation is required. The Optical Stimuli Generator is constantly sending light to the cells in the form of a 608x684 pixel images projection. We define as optical stimuli whatever variation in the image, from a single pixel to the whole frame. As already anticipated in Chapter 3, two different ways are possible to change the projected image. The fastest way is to change the settings of the HDMI Controller to point to a different location in the DDR Memory, where the new image to be projected was previously written. In this case, the additional latency is due to the refresh rate of the projector, which runs at 60 Hz, thus in the worst case it is:

$$\text{Projector}_{\text{refresh time}} = \frac{1}{60\,\text{Hz}} = 16.667\,\text{ms} \tag{5.2}$$

The additional delay to change the settings are negligible compared to this one. The limitation of this approach is that the images must be ready before the stimulation decision. This can be achieved or via an efficient software pipeline, that prepares the images for the next stimuli in advance, or by storing all the possible stimuli patterns in the DDR in an initialization stage.

If none of these solutions can be adopted, then the only possible way is to write the new image to be projected at runtime. This is done through the Image Creator module that is able to draw rectangular blocks with different size, shape, color or starting point, upon receiving the right command from the ARM processor. In this case, the additional latency linearly depends on the number of pixels to be changed, as reported in Fig. 5.5. The average time to change a pixel is 6.55 ns. However, if the whole frame needs to be changed, an optimized hardware mechanism allows to do that in slightly less than 2.5 ms (6 ns/pixel). A summary of the different closed-loop latency is reported in Table 5.7.

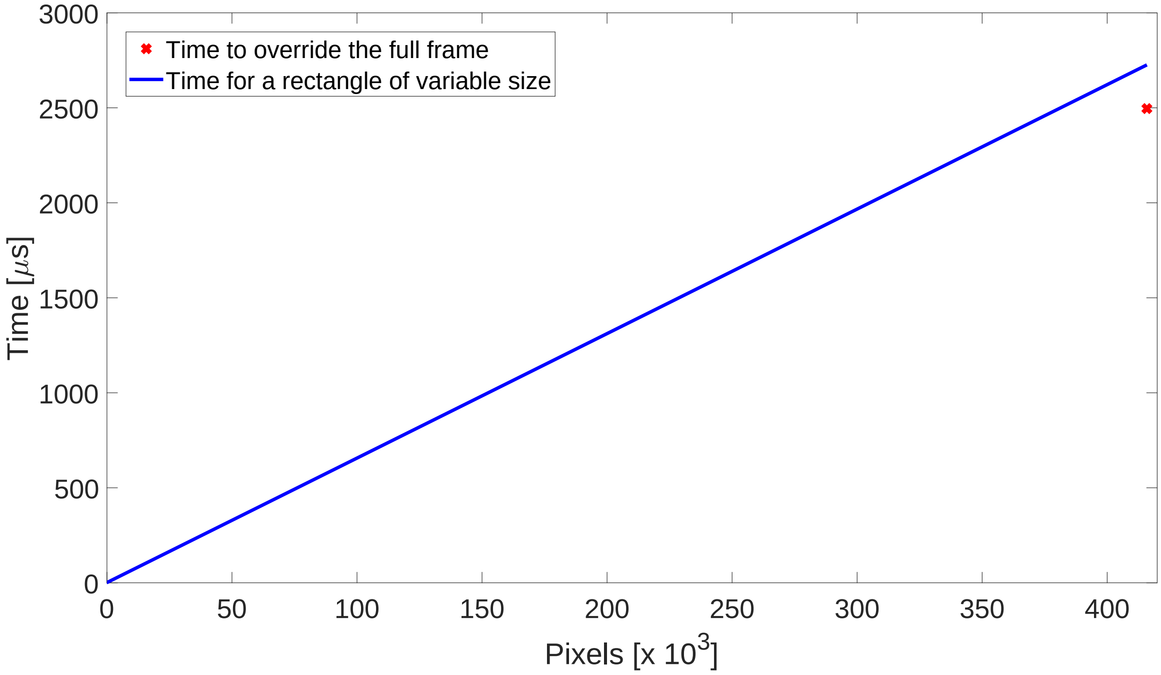


Figure 5.5 Time to override the image pixels.

## 5.6.1 Digital Signal Processor comparison

To point out the superiority of the APSoC device for this kind of applications involving massive processing and very low latency, I studied the feasibility of the filtering step in a Digital Signal Processor. The device at my disposal was the OMAP-L138 from Texas Instruments featuring a C674x DSP core and an ARM processor. An optimized version of the 64-tap FIR filter was coded in the DSP core. What I observed is that the execution time is highly dependent on how many samples of the same channel are being processed altogether. Indeed, buffering the data and then executing the FIR filtering on longer signals leads to a much lower execution time per sample. This is due to the reduction in the overhead required,

Table 5.7 Latency Evaluation, times in ms.

| Task | Electrical Stimuli | Optical Stimuli (worst case) | |
|---|---|---|---|
| | | Method 1 | Method 2 |
| Processing | 1.87 | 1.87 | 1.87 |
| Electrical stimulation | 0.001 | - | - |
| Image Creator | - | - | 2.5 |
| Projector$_{\text{refresh time}}$ | - | 16.67 | 16.67 |
| Total | 1.87 | 18.54 | 21.04 |

the data necessary for the filtering of a channel is loaded only one time for the processing of multiple channels. However, to wait for the buffer to be filled means to add a considerable latency to the processing. In fact, the second channel to be filtered has to wait until all the samples of the first channel are filtered. These results are reported in Table 5.8. Now if we consider a very short buffer size (4 samples) to limit as much as possible the additional latency, we can calculate the final throughput offered by the 456 MHz DSP as:

$$DSP_{throughput} = \frac{DSP_{clock}}{56.5\,\frac{\text{cycles}}{\text{sample}} \cdot \frac{f_s}{\text{channel}}} = \frac{456\,\text{MHz}}{56.5\,\frac{\text{cycles}}{\text{sample}} \cdot \frac{18\,\text{kHz}}{\text{channel}}} = 448 \text{ channel} \qquad (5.3)$$

Table 5.8 Performance analysis of 64-tap FIR filter implemented on a Digital Signal Processor for different buffer sizes. The additional latency is calculated as $\frac{\text{buffer size}}{f_s}$ with $f_s = 18\,\text{kHz}$, the percentage value is related to the final latency found in the APSoC of 1872 µs.

| Buffer size | Clock Cycles | Additional Latency | |
|---|---|---|---|
| [# samples] | per sample | [µs] | [%] |
| ∞ | 16 | - | - |
| 1024 | 18.71 | 56889 | 3039 |
| 128 | 19.79 | 7111 | 380 |
| 64 | 21.00 | 3556 | 190 |
| 32 | 23.31 | 1778 | 95 |
| 16 | 28.06 | 889 | 47 |
| 8 | 37.75 | 444 | 24 |
| 4 | 56.50 | 222 | 12 |

This is the maximum number of channels that can be processed in real-time with the C674x. However, the DSP used was obsolete, so the result must be scaled up to a state-of-the-art device in the same price range of the used APSoC. One such device can be the Texas Instruments 66AK2E05 DSP, which holds ten times the performances of the C674x thus can, in theory, perform the filtering stage on all the 4096 input channels. The advantage of using the 66AK2E05 DSP for the filtering is that it takes less time to be implemented compared to doing it in an FPGA. However, the disadvantages are manifold:

- Increased processing latency of 12%.
- No much processing time remaining for the spike detection and the dynamic threshold update.
- No other resources available for all the additional blocks implemented on the Zyon.
- No possibility to scale up the design or add additional features in the same device.

## 5.7 SNR improvement evaluation

As already stated in the previous chapter, the hardware for the noise reduction was implemented in a different chip, the Intel® Arria® 10 FPGA, more specifically the model SX 270. This was not a design choice but forced merely by the availability of devices in the 3·Brain company. Even if the rate of samples to process is extremely elevated, $1.28 \cdot 10^9$ Samples/s, thanks to the simplicity of the adopted algorithm and the design optimizations aimed at reducing the computational complexity, the resulting hardware resources needed for the denoising are quite limited. The final consumption figures, reported by the Intel Quartus Prime software used for the implementation, are reported in Table 5.9.

The denoising part has not been merged yet with the Zyon; however, we can empirically verify whether the resources needed can fit the remaining ones available in the Zynq-7020

Table 5.9 Resources usage for the noise reduction.

| | Hardware resources | | |
|---|---|---|---|
| | Logic Elements | Memory k bits | DSP |
| Noise Reduction | 28253 | 115 | 0 |
| Arria 10 SX 270 | 270000 | 15000 | 1660 |
| Usage | 10.5% | 0.8% | 0% |

and reported in the Table 5.6. The first thing to do is to convert the resources used in the Intel FPGA platform into the respective Xilinx one. The Logic Elements (LEs), reported in the Table 5.9 and featured in the Intel boards, are basically composed by a four-input look-up table, plus some other small glue logic block. In the Xilinx devices, instead, the integrated look-up tables can support six input each. Under the assumption that the design and the implementation software are able to efficiently exploit the larger LUTs, the conversion becomes 1.5 LE = 1 LUT. Now, about the Memory, each BRAM block in the Z-7020 is composed by two memories of 18 kbits each. Thus only 3 full BRAM plus one smaller memory is enough. Finally, for the DSP we can consider an exchange ratio of 1 to 1, which is not 100% true, but it does not matter since no DSPs are actually used. The final estimated results are reported in the Table 5.10. As can be seen, everything fits the Z-7020 with even a margin that can eventually compensate the approximation done with the conversion of hardware resources from Intel to Xilinx.
As of now, the denoising system implemented in the Arria® 10 FPGA is under test in the real environment. If the same results obtained with the offline simulations hold true, it means that an SNR improvement of 16 can be achieved by giving up half of the sampling frequency per electrode. This is achieved by reducing the input referred noise of the BioCam X by a factor of 4.

Table 5.10 Total resources estimation adding the noise reduction to the Zyon.

| | Hardware resources | | |
|---|---|---|---|
| | Logic Elements | Memory k bits | DSP |
| Noise Reduction | 28253 | 115 | 0 |
| Conversion | 1.5 : 1 | 36 : 1 | 1 : 1 |
| | LUT | BRAM | DSP |
| Noise Reduction | 18835 | 3.5 | 0 |
| Current Zyon | 30896 | 122.5 | 65 |
| Total | 49731 | 126 | 65 |
| Z-7020 | 53200 | 140 | 220 |
| Usage | 93.5% | 90.0% | 29.5% |

# Chapter 6

# *Ex vivo* experimental results

*"Life is what we make of it. Travel is the traveler.*
*What we see is not what we see but what we are."*
Translated from *Livro do Desassossego* by Fernando Pessoa (poet, writer, philosopher)

## 6.1 Render to Caesar the things that are Caesar's

All the experiments that will be described here were performed at the NetS$^3$ Lab, at the Italian Institute of Technology in Genoa, in collaboration with the people working there. I want to acknowledge especially the contributions of: Gian Nicola Angotzi that helped me to make the setup up running; Fabio Boi who was the expert in the preparation of the biological tissues; Sara Zaher and Davide Lonardoni who significantly contributed to the last experiment that will be presented here; and finally Luca Berdondini for his guidance throughout the whole project.
The chapter is organized as follows: The next section describes the preparation of the retinas used for the experimentation. Sections 6.3 and 6.4 describe the experiments performed to assess the functionalities of the Zyon system, i.e., signal acquisition, real-time processing, detection of neural activity, and closed-loop stimulation. Finally, the last section 6.5 deals with biologically relevant experimentation aimed at classifying automatically retinal ganglion cells based on their behavior.

## 6.2 *Ex vivo* retinas

The experimentation was performed using retinas whole-mounts from C57BL/6 male mice exposed to controlled optical stimulation.

### 6.2.1 Retinas preparation

The protocols followed for the preparation of the retinas are similar to the ones described in (Hilgen et al., 2017a). The mice were adapted to the dark for twelve hours before the execution. The sacrifice was performed by cervical dislocation after anesthesia obtained via $CO_2$. The eyeballs of the dead mice were enucleated, and the cornea, the lens, and the vitreous body were removed in order to isolate the retinal tissues. After that, the retina was placed in the Arena HDMEA from 3·Brain that was preconditioned with Neurobasal at 37 °C for two hours. The retinal ganglion cell (RGC) layer of the retina was faced down in contact with the electrodes for the acquisition of the electrical signals, while the photoreceptor layer faced up and exposed to the light from the projector. In order to keep the retina still throughout the whole experiment, a polyester filter from Sterlitech Corp. and a circular anchor were placed on the photoreceptor layer. Moreover, to maintain the tissue vital, a constant flow of AMES media from Sigma-Aldrick (now part of the Merck Group) was supplied through a peristaltic pump (~1 ml/min) into the well containing the HDMEA. The media was supplemented with sodium bicarbonate (1.9 g/l) and equilibrated with carboxigen (95% $O_2$ and 5% $CO_2$). A picture of a retina placed on the Arena HDMEA is provided in Fig. 6.1.

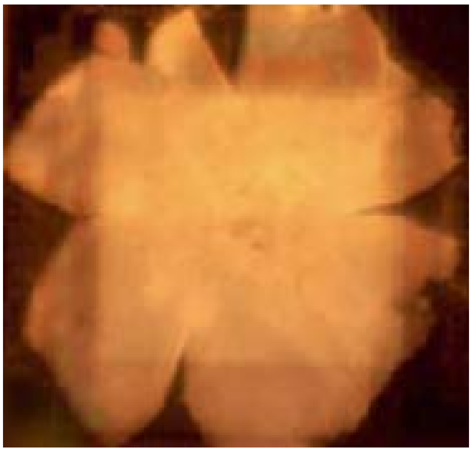


Figure 6.1 Retina on the Arena HDMEA chip.

### 6.2.2 Ethical statement

All the experiments on ex vivo retinas were performed in accordance with the guidelines established by the European Community Council (Directive 2010/63/EU of 22 September 2010). All procedures involving experimental animals were approved by the institutional IIT Ethics Committee and by the Italian Ministry of Health and Animal Care (Authorization number 110/2014-PR, December 19, 2014).

## 6.3 Open-loop system evaluation

The first experiment to be discussed here was performed in an open-loop mode, without the execution of any closed-loop algorithm or generation of stimuli. In this condition, the Acquisition unit records only the spontaneous neural activity of retinal ganglion cells. This was the first stage of validation to assess the capability of the Zyon in acquiring and filtering the input signals, and at the same time detecting spikes from all the channels in real-time. The ARM core instead of closing the loop was in charge of extracting and saving the waveforms of the detected spikes. To evaluate the performances of the real-time spike detection, I considered as the ground truth the results produced by the BrainWave X software from 3·Brain. The software processed offline all the acquired raw signals with a hard threshold algorithm. Both the static and the dynamic threshold mode implemented on the Zyon were tested.

For the static threshold, to make the comparison fair, the same parameters were used for the real-time hardware and the BrainWave X software: threshold set at $-200\,\mu\text{V}$ and refractory period of $2\,\text{ms}$. Thanks to the Time Synchronization signal produced by the Zyon, it was possible to accurately compare the timestamps of the spikes found online with those found offline. With a tolerance of $\pm 500\,\mu\text{s}$ the coincidence of spikes in the 4096 channels was over 99%. A graphical representation of the online-offline comparison from a single channel is depicted in Fig. 6.2.

For the dynamic threshold, a direct comparison with the results obtained offline with the BrainWave X software was not so easy. It would have been necessary to know the true nature of every single threshold crossing obtained with one of the detection but not with the other, thus a massive manual work to discriminate real spikes from noise peaks. A more superficial visual work was performed, confirming that the dynamic threshold correctly detects spike events, ignoring the background noise, as represented in Fig. 6.3.

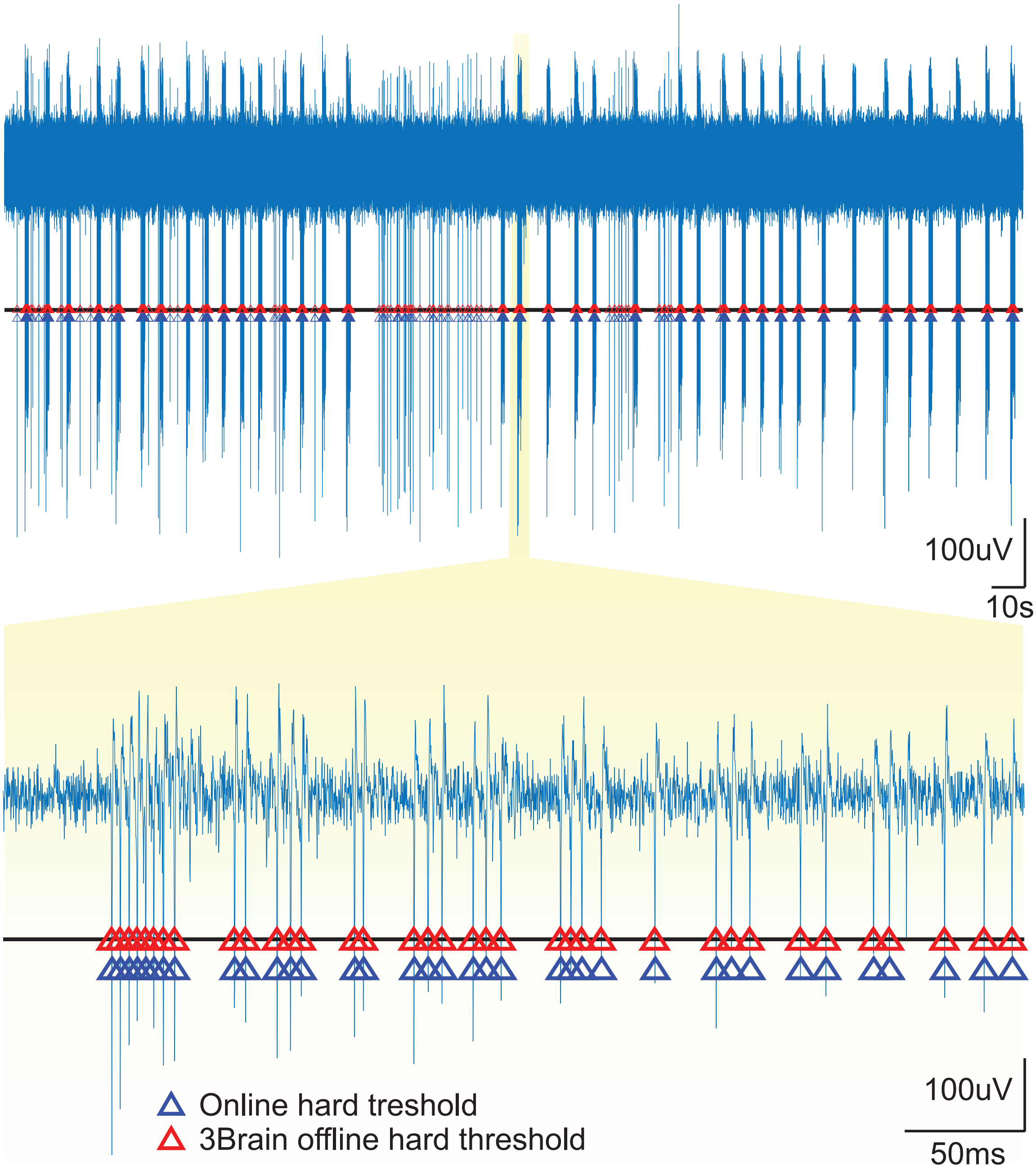


Figure 6.2 Comparison between Online and Offline spike detection performed with static Threshold (black line). The blue triangles are the spikes detected by the real-time hardware, while the red ones are the ones detected offline by the BrainWave X software algorithm. The raw data represents the basal neural activity of retinal ganglion cells recorded by one of the 4096 electrodes of the HDMEA.

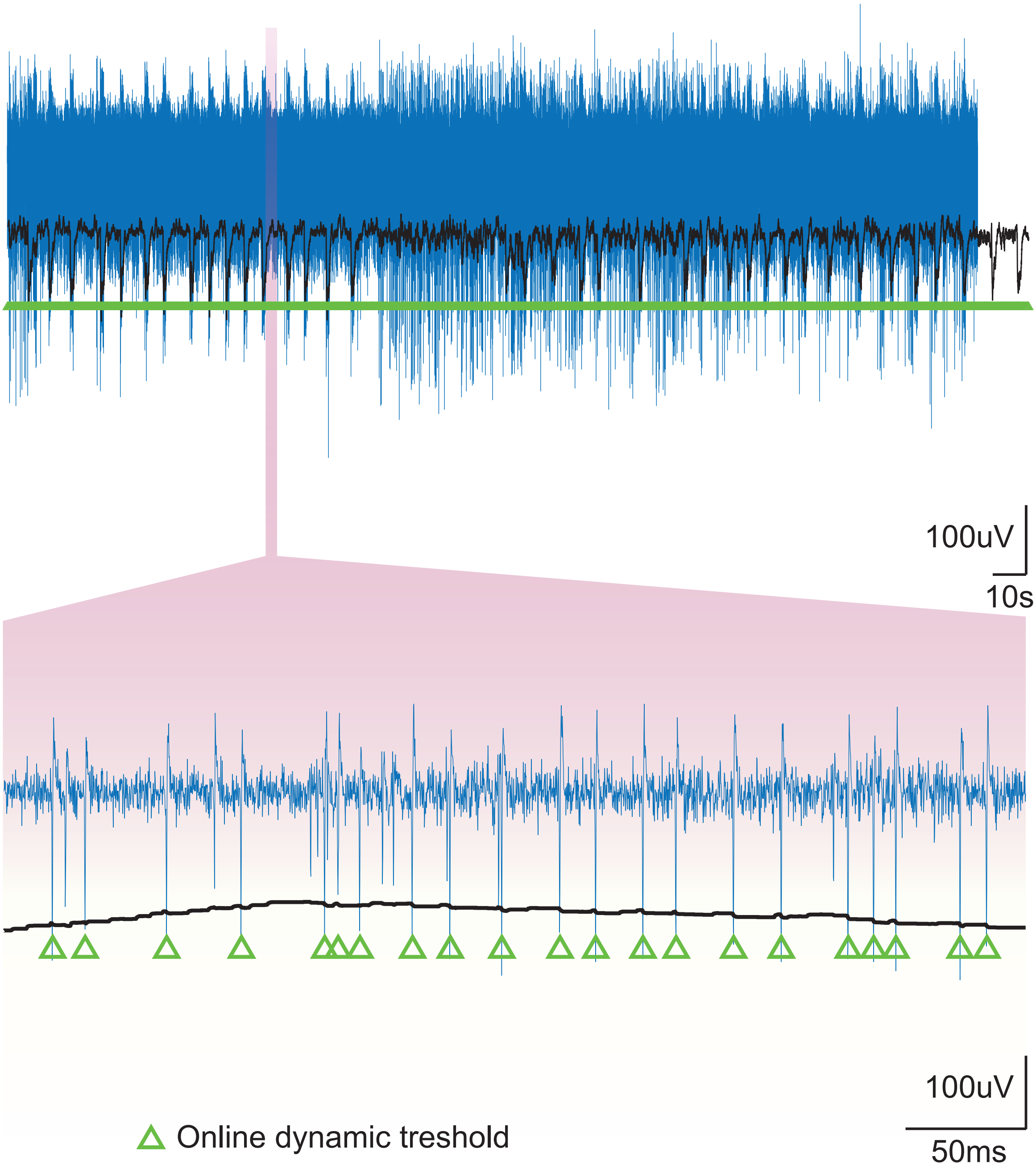


Figure 6.3 Example of spike detection performed with the dynamic threshold (black line). The local standard deviation of the signal is calculated using a sliding window mechanism and then used to update the threshold as explained in Chapter 3. The detected spikes are marked with green triangles. As for the previous image, the detection was done on the basal activity of retinal ganglion cells.

## 6.4 One light, one mind, flashing in the dark

Only a Minority of you will get the title reference, but that is ok.
The second experiment was focused on the validation of the closed-loop capabilities of the system. This time the retina was subjected to the optical stimuli delivered by the DLP® LightCrafter™ Evaluation Module controlled by the Zyon. The closed-loop algorithm, running on the bare-metal core, takes the stimulation decision based on the overall mean firing rate (MFR) calculated on a time window of 50 ms. The processing steps of the algorithm are reported as a pseudo-code in the Listing 6.1. For every new input frame written in the DDR Memory, the processor analyzes all the channels storing the information about the spike presence (channel ID and time stamp) for verification purpose. It also calculates the total spike occurrences in the whole array; this value is then used to update the MFR together with the last value of the previous window. Each time the MFR is below a certain programmable threshold (MFR_THR in the listing), 1 s of light stimulation is sent to the

Listing 6.1 Closed-loop code.

```
for each new frame
{
    spikes = 0
    for each channel
    {
        if there is a spike
        {
            store channel_id and time
            spikes++
        }
    }
    MFR = MFR + spikes - spikes_memory[addr]
    spikes_memory[addr] = spikes

    if(MFR < MFR_THR)
    {
        store time
        send stimulus
    }

    if(addr == (WINDOW_SIZE-1))
        addr = 0
    else
        addr++
}
```

cells in the form of a completely white frame (luminous intensity $0.22\,\mathrm{cd/m^2}$). The temporal information of each stimulus is stored as well. In the last part of the listing, we have the update of the address used to point to the last spikes count in the window. The update is based on the WINDOW_SIZE parameter, which can be calculated as the window duration in seconds multiplied by the sampling frequency in Hertz. The Fig.6.4 is a good example of one experimental session performed with the described algorithm. The top of the figure holds the raster plot depicting the occurrence of spikes (blue asterisks), the stimulation decision (red triangles) and also the optical stimulation lasting 1 s (yellow area). The bottom of the figure represents instead the trend of the MFR (green line), which triggers a stimulation each time it goes under the threshold (red line), here fixed at 15 spikes. Around 100 ms after the beginning of each stimulation and for the whole duration of it, a strong neural response is recorded. This confirms what already known in literature (Pang et al., 2003), and demonstrates the proper functioning of the developed system.

One last discussion on the algorithm must be made regarding its execution time. In order to operate correctly in real-time, the processing of each frame must be made in less than $T_s = 1/f_s = 55.56\,\mu\mathrm{s}$, which corresponds to 37000 ARM clock cycles. Two different modes of the algorithm presented above have been tested, one executing only the steps required for the stimulation decision, while the other also storing the information on all the detected spikes. In the first mode, the execution time did not change throughout all the experiment, remaining constant around 30000 clock cycles (45 µs), thus always satisfying the real-time constraint. For the second mode, instead, the execution time heavily depends on the number of detected spikes. This is due to the introduced overhead of saving the channel identifier and temporal tag for each spike in the external memory (required later for offline verification). The maximum number of spikes that can be processed with this version, without violating the real-time constraint, is around 150, which leads to an execution time close to 37000 clock cycles. In case that a higher number of spikes is detected in a single frame, the constraint is no more respected, with the worst case being 4096 spikes detected simultaneously that leads to an execution time of 74000 clock cycles (111 µs). However, such cases are very unlikely, they can happen only once every many iterations if we consider the normal behavior of neurons, that present a refractory period around 2 ms after each spike (Seymour et al., 2017). The last thing to do is to evaluate if the real-time constraint is met in the long run. The algorithm is able to process 150 spikes per frame which translates in 2.7 million spikes/s. Dividing by the number of channels we have roughly 660 spikes/s per channel, much larger than the maximum firing rate of a neuron that is limited to 400-500 spikes/s (Harris et al., 2012). This proves that the closed loop-algorithm is able to comply with the real-time constraints even if one iteration takes more than the available clock cycles. Moreover, since the processing time does not depend on the size of the sliding window, any value can be chosen for such parameter. In the extreme case of a window of only one sample, the MFR becomes the instantaneous firing rate (IFR).

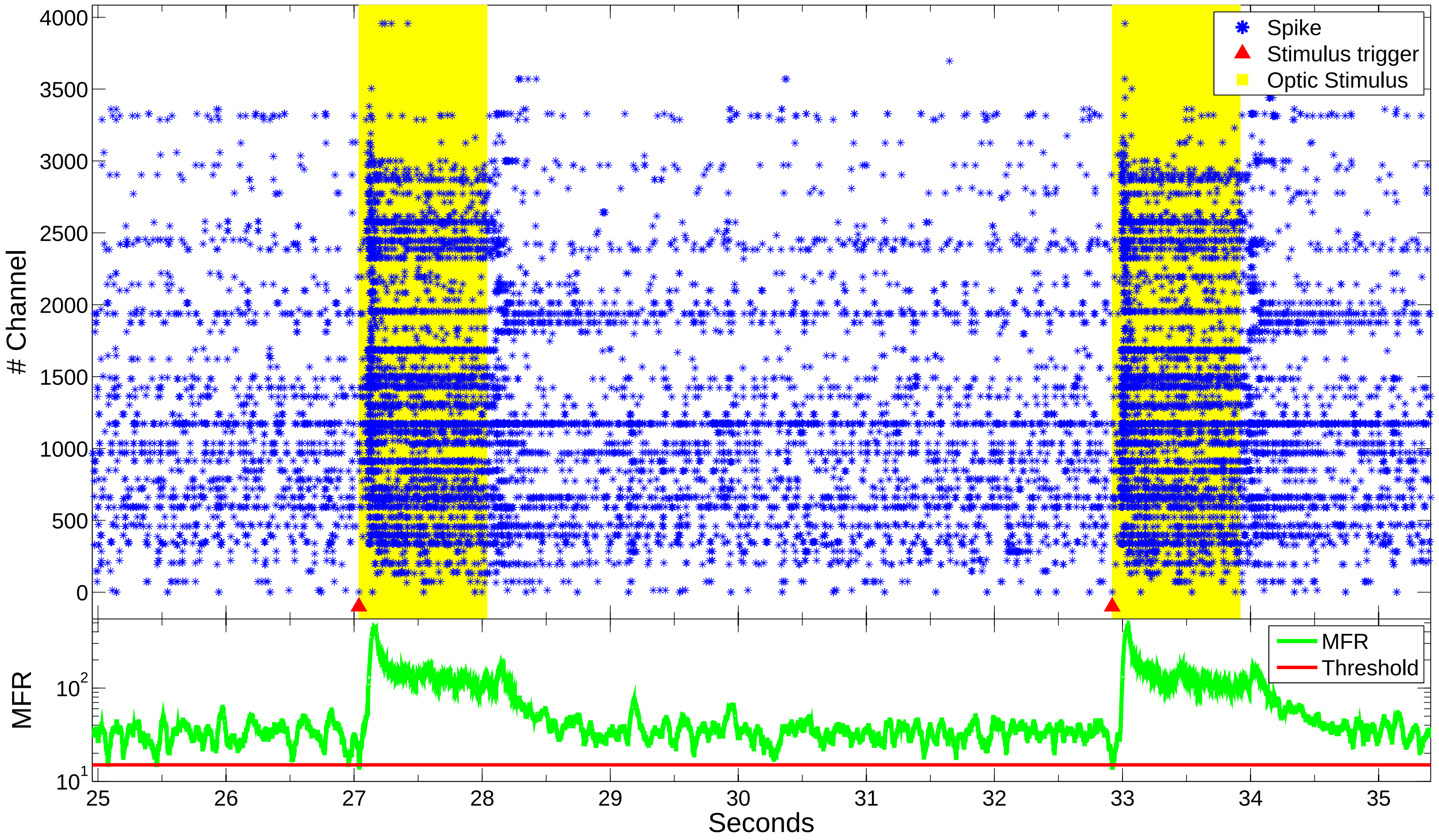


Figure 6.4 Closed-loop experiment realized on a mouse retina whole mount subjected to optical stimulation. The closed-loop algorithm calculated the mean firing rate on a window of 50 ms, whenever it went under a certain threshold, an optical stimulus was sent to the cells drastically increasing their firing rate.

## 6.5 ON and OFF RGC classification

The last experiment to be presented is composed of two parts: data reduction and ON-OFF response classification. Since the electrode density of HDMEAs is so high, it is common that the same spikes are recorded by multiple neighboring microelectrodes. The first part is aimed at reducing the data to elaborate by selecting only the unique channels from the whole array. The second part takes advantage of the data reduction to speed up the classification of the retinal ganglion cells on the basis of their behavior. A total of 5 tests with different retinas have been performed with the experimental protocol that will be described in the following. Results are reported as the mean ± standard error of the mean (SEM) among all the tests defined as:

$$SEM = \frac{\sigma}{\sqrt{n}} \tag{6.1}$$

where $\sigma$ is the standard deviation and $n$ is the number of observations, 5 in our case.

### 6.5.1 Data Reduction

Following a protocol similar to the one reported in (Hilgen et al., 2017b), the goal is to cluster the activity of unique units discarding the duplicates. In order to do so, the retinal basal activity was recorded while the photoreceptor layer was subjected to an isoluminant gray full-field stimulus (luminous intensity $0.11\,\mathrm{cd/m^2}$). The detected sub-millisecond correlated (SMC) spikes detected were exploited to discriminate the unique units.

The protocol is composed of different stages. First, all the channels with a firing rate inferior to $0.5\,\mathrm{spikes/s}$ are considered inactive and discarded. All the other channels, from now on the active electrodes (AEs), are kept. The next step is to detect the sub-millisecond correlated spike-trains. In order to do that, a similarity index S has been calculated, quantifying how much a spike-train $sp_1$ from one AE is contained in another one $sp_2$ from a different AE. The following equation has been used:

$$S(sp_1,\ sp_2) = \frac{\sum_{i,j}^{N_1,N_2} \delta_{tol}(sp_{1,i},\ sp_{2,j})}{min(N_1,N_2)} \tag{6.2}$$

where $N_1$ and $N_2$ are the respective number of spikes in each spike-train, while $sp_{1,i}$ is the time stamp of the $i$-th spike of $sp_1$. The indicator function $\delta_{tol}$ is instead defined as:

$$\delta_{tol}(sp_{1,i},\ sp_{2,j}) = \begin{cases} 1, & |sp_{1,i}, sp_{2,j}| \leq tolerance \\ 0, & |sp_{1,i}, sp_{2,j}| > tolerance \end{cases} \tag{6.3}$$

The *tolerance* parameter is added to take into account the small jittering in the detection time for the different channels. It was set to be equal to the sampling period ($T_s = 1/f_s$), so two spikes of neighboring electrodes are considered the same only if they are detected in the same frame or in two subsequent frames. The similarity index function, $S$, gives a result between 0 and 1, where 0 means that the spike-trains are totally different and 1 that they are identical. It was calculated for each active electrode with respect to its neighboring active electrodes (nAE), meaning a total of 8 computation if all are active. The similarity with respect to inactive electrodes and out of the grid electrodes (for the ones at the border of the chip) was set to 0 without any calculation. The results were stored in a matrix 8x4098. A graphical representation of the results from one of the tests is given in Fig. 6.5.

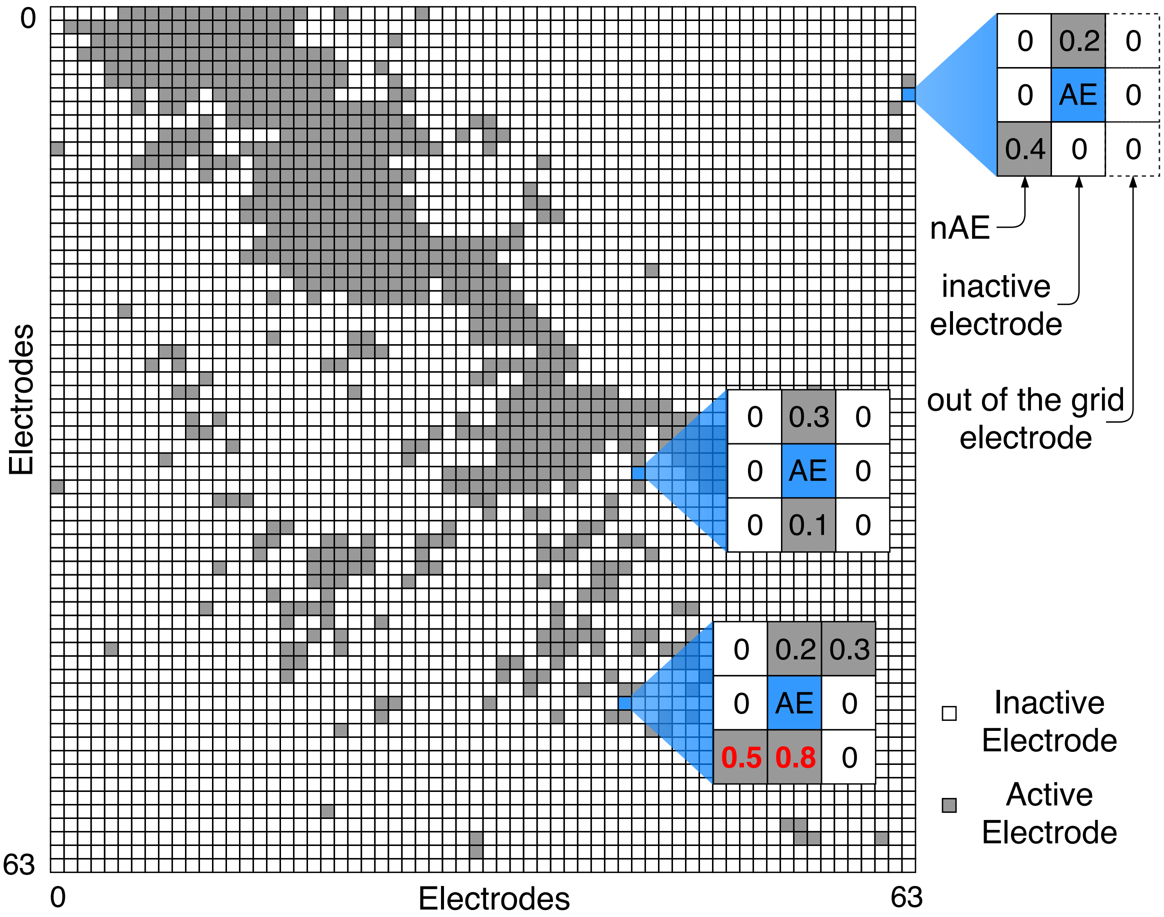


Figure 6.5 HDMEA with distinction between active electrodes (grey squares) and inactive electrodes (white squares). The similarity index is calculated for each active electrode (AE) with respect to its neighboring active electrodes (nAE). The red text indicates that the similarity index is above the 0.4 threshold.

The next step to perform is to cluster the coincident spiking activity from different nAEs, which is most likely coming from the same neuron. In order to do so, the protocol involves two thresholding phases to be performed on the similarity matrix just created. The first phase discards all the electrodes that do not have any similarity index above a certain threshold, fixed at 0.4. In the Fig. 6.5 the threshold crossing is emphasized with the red text as can be seen in the bottom-most highlighted AE. The other two AE highlighted in the figure have all the similarity index below the threshold and are discarded for the next phase. In the second phase, a local threshold (LT) is calculated for all the AEs that passed the first thresholding. This new threshold which is unique for each AE is computed as the mean plus the standard deviation of the nAE similarity indexes. Being $\mathrm{LT_A}$ and $\mathrm{LT_B}$ the local threshold of two neighboring electrodes, A and B respectively, then they are clustered together if their similarity index $S(\mathrm{A, B})$ is above both $\mathrm{LT_A}$ and $\mathrm{LT_B}$. If A or B is already a member of a cluster, then the other one is added to the same cluster as well. With this second thresholding phase, each electrode is assigned to the cluster that matches the best. Each AE that is not merged with any other, thus also the ones discarded in the first thresholding phase, is considered a single unit cluster, and its electrode is defined as the leader electrode (LE). As a last step, for each multi-unit cluster, the electrode with the highest firing rate is selected as the LE, the others are discarded.

This clustering part is performed continuously with iterative steps until the incremental variation met the stopping criterion. At every second of recording the similarity matrix and the leader electrodes are updated with the new data. The incremental variation between the current step $k$ and the previous step $k-1$ is calculated, considering all the clusters, with the following equation:

$$variation(k) = \frac{\sum_i \delta\Big(LE_i(k),\ LE_i(k-1)\Big)}{N}\ x\ 100 \tag{6.4}$$

where $N$ is the number of AEs and $\delta$ is the Kronecker delta function defined as:

$$\delta\Big(LE_i(k),\ LE_i(k-1)\Big) = \begin{cases} 1, & LE_i(k) = LE_i(k-1) \\ 0, & LE_i(k) \neq LE_i(k-1) \end{cases} \tag{6.5}$$

while $LE_i(k)$ and $LE_i(k-1)$ identify the $i$-th leader electrode at the iteration $k$ and $k-1$ respectively. The stop criterion is met when $variation(k)$ remains under 0.25% for three successive iterations. For example, if $N$ is around 4000 (i.e., almost every electrode is a cluster on its own), then the maximum mismatch allowed is 10 electrodes, for $N$ close to 2000 instead it is 5. When the stop criterion is met, the first part of the experiment is concluded, for the next part only the leader electrodes determined in the last iteration are considered, minimizing computational and memory needs.

Finally, a graphical representation of the multi-unit clusters from one test is reported in Fig. 6.6. The SMC spiking activity leads to the formation of clusters of various size and shape, suggesting that more electrodes are sensing from the same neuron or group of neurons. In each test performed, this procedure detected SMC nAEs right after the first seconds of recording. Their number stabilized at around ~25% of the total active electrodes after a short phase of significant decrease. The stopping criterion was met after 97±30 s in the various tests. With this procedure, it was obtained an average reduction in the number of channels to process of 26±10% of the total number of AEs, reached already after a few tenths of seconds after the beginning of the recording.

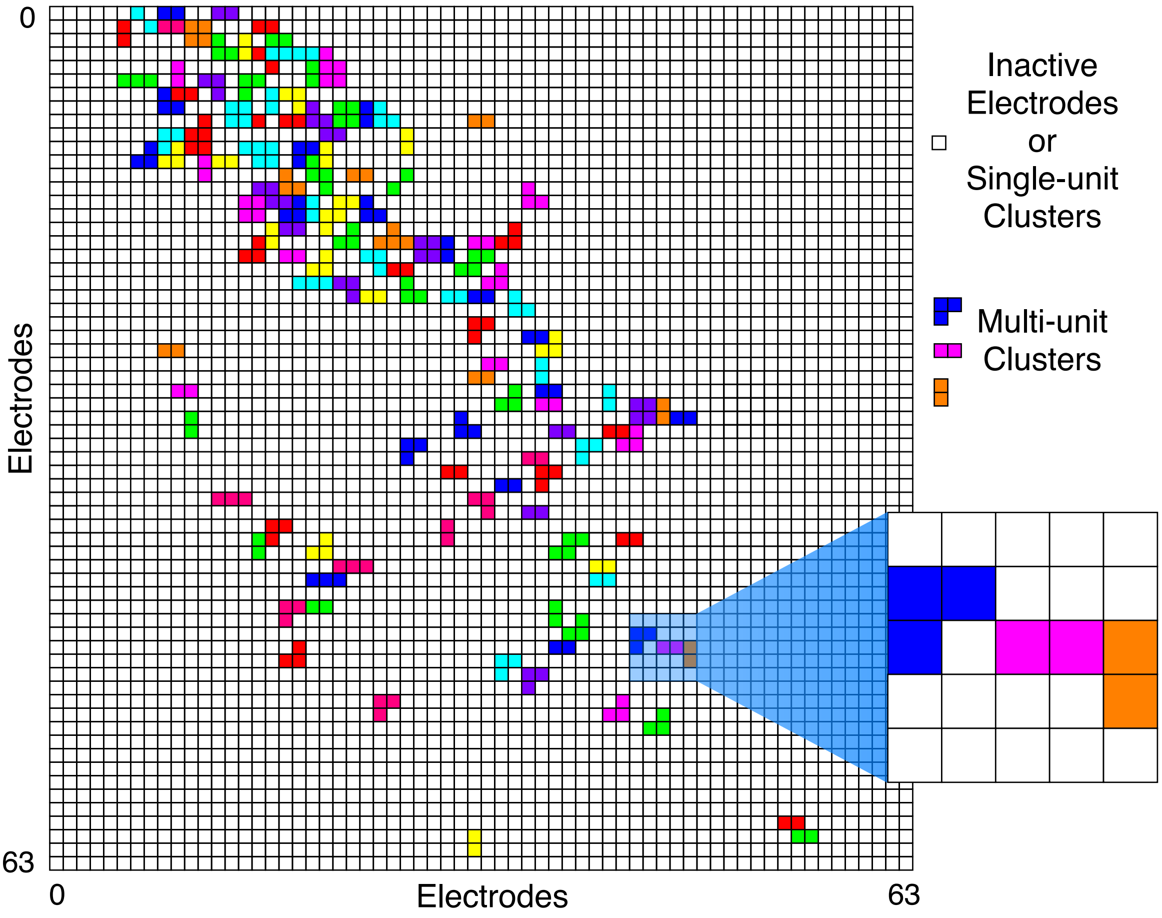


Figure 6.6 Multi-unit clusters of coincident activity detected at the stop criterion.

### 6.5.2 ON and OFF response classification

The goal of the second part of the experiment is to classify the retinal ganglion cells according to their behavior. When an RGC increases its mean firing rate, with respect to the basal activity, as a response to a white stimulus, and decreases it as a response to a black stimulus, then it is classified as ON-cell. Vice versa, if the MFR increases under a black stimulus, and decreases with a white one, then the RGC is classified as an OFF-cell. The data reduction protocol of the first part clustered each RGC and selected, for each cluster, the leader electrode; now each LE is classified as either "1" or "0" whether it holds an ON or OFF kind of behavior. Basically, the *i*-th electrode is classified as $C_i$ accordingly to the following equation:

$$C_i = \begin{cases} 1, & (MFR_{i,w} > MFR_{i,g}) \text{ AND } (MFR_{i,b} < MFR_{i,g}) \\ 0, & (MFR_{i,w} < MFR_{i,g}) \text{ AND } (MFR_{i,b} > MFR_{i,g}) \end{cases} \tag{6.6}$$

where $MFR_{i,w}$, $MFR_{i,b}$ and $MFR_{i,g}$ are the mean firing rate of the *i*-th LE under white, black or gray isoluminant stimulation. Note that some electrodes may end up unclassified if neither of the conditions for $C_i$ is met. To achieve this classification, the Zyon unit is in charge of automatically running a stimulation protocol that subject the retina to full-field flashes of black (luminous intensity $0\,\mathrm{cd/m^2}$) and white ( $0.22\,\mathrm{cd/m^2}$) frames. The basal activity of each LE, under grey stimulation ($0.11\,\mathrm{cd/m^2}$), was determined in the previous part of the experiment. During the stimulation protocol, the Zyon is also in charge of analyzing the spiking activity of all the leader electrodes. As for the clustering part, also the classification of ON and OFF RGCs is done iteratively. Subsequent projection of black and white stimuli are provided, while the $C_i$ is continuously updated at each iteration with the new spiking information. The stop criterion is defined with the help of a switch function, that computes the differences in the classification of LE, and is defined as:

$$switch(k) = \frac{\sum_i |C_i(k) - C_i(k-1)|}{N} \, x \, 100 \tag{6.7}$$

Basically, the function calculate the percentage of total transitions, from ON- to OFF-type and OFF- to ON-type classification, of the LE in two consecutive iterations. If $switch(k)$ is under 3% for three subsequent iterations, then the protocol can stop, and the last classification $C_i$ is considered as the best one. In Fig. 6.7 is reported the trend of the $switch(k)$ function, red line, for the various tests. As it can be seen, the stop criterion is met after few iterations of the protocol. The boxplot in the bottom of the figure reports the exact distribution of the number of iterations required, 10.2±3.0. In each iteration, the black and white flashes last

1 s each. With all the further iterations after the convergence point, the *switch*$(k)$ function always stays under the threshold, red dashed line, and decreases even more. The blue line represents instead the variation of switches in the subsequent iterations. It can be seen that, after the convergence period, it stays stable close to 0%.

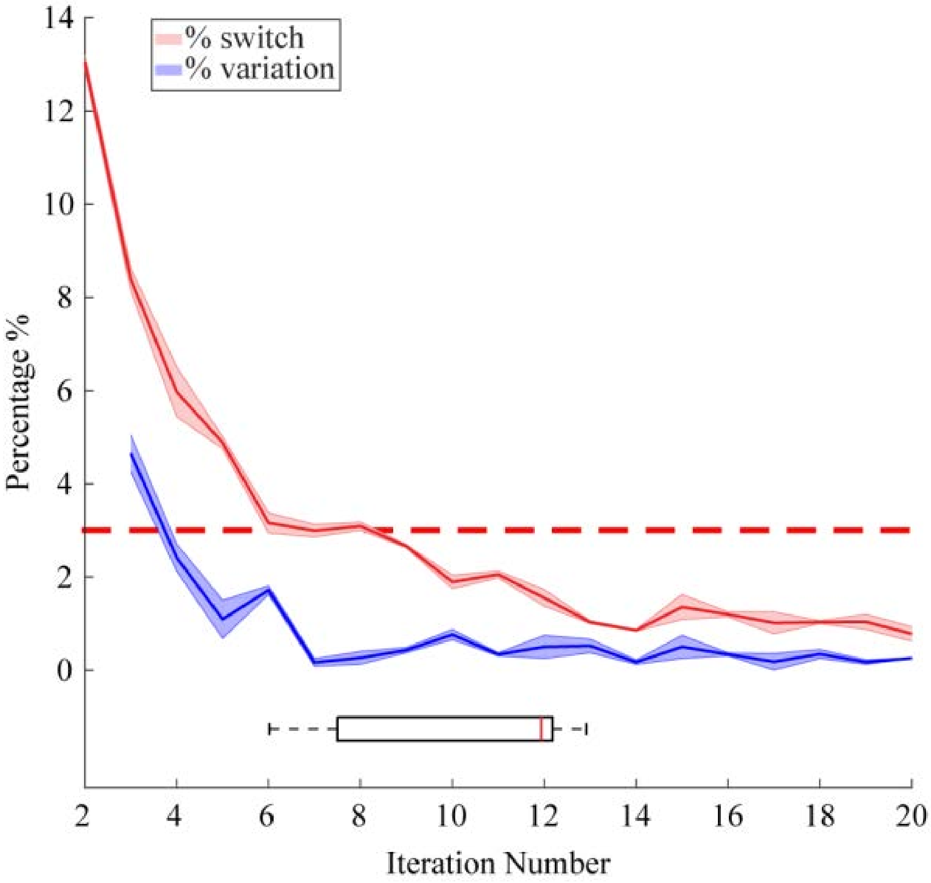


Figure 6.7 Trend of the *switch*$(k)$ function for the various tests (red line) and its variation (blue line) on subsequent iterations. The boxplot in the bottom represents the distribution of the number of interactions required to reach the stop-criterion, *switch*$(k)$ function below the threshold (dashed red line) for 3 subsequent iterations. Shaded area refer to ±SEM.

The protocol adopted was able to classify 90.8±4.5% of the total number of LEs as either ON- or OFF-type. The unclassified LEs were visually inspected. An ambiguous response was noticed in 11% of them, while the others presented a poor light-evoked activity (i.e., mean firing rate 0.9±1.0 Hz. The automatic protocol rightfully excluded such behavior from the classification, since it belongs to unrecognizable neural activity.

# Chapter 7

# Conclusions

*"Educate yourselves because we will need all our intelligence.*
*Agitate yourselves because we will need all our enthusiasm.*
*Organize yourselves because we will need all our strength."*
Antonio Gramsci

## 7.1 Research contributions

The central achievement of this PhD is the design and implementation of a real-time neural signal processing architecture for closed-loop experiments. The main research contributions are listed in the following:

- This work is the first to exploit an All Programmable System on Chip (APSoC) device for application of neural interface. An APSoC is a modern heterogeneous FPGA integrating in the same chip not only the typical programmable logic but also hardwired processors. Thanks to the software programmability, a high degree of freedom is ensured in such devices. Moreover, the programmable logic features hardwired blocks, such as DSP slices and BRAMs, optimized for specific tasks, allowing the device to achieve very high elaboration performances. The overall processing required for the application has been split among the available resources, exploiting the heterogeneity of the device, to achieve high throughput and flexibility at the same time. The computational intensive tasks, such as the band-pass filtering and the spike detection, have been implemented in the programmable logic, while the closed-loop stimulation decision is taken by the software running in the processor.

- A deep design space exploration has been performed to exploit all the degrees of freedom offered by both the neural application and the processing architecture. Thanks to the high degree of reconfigurability, the processing system can target different experimental setups and different performance requirements. The exploration was performed targeting three use cases, each with a different number of input channels to process and a different sampling frequency. Parameters, such as the order of the band-pass filters, have been tuned for each use case to obtain the best trade-off between accuracy of the spike detection, resources consumption, and elaboration latency. It was demonstrated that the resources consumption for each use cases fit a low-cost APSoC from the Xilinx Zynq®-7000 SoC family.

- A working prototype has been implemented in a ZedBoard by Avnet featuring a Xilinx Zynq®-7020. Compared to the state-of-the-art closed-loop systems, the number of parallel processed channels in the developed prototype is more than one order of magnitude higher, 4096 in total. In the meanwhile, the processing latency is kept under 2 ms allowing for a wide range of closed-loop experiments.

- A complete experimental setup has been assembled for closed-loop *in vitro* and *ex vivo* experimentation. The interface with the biological cells is achieved through an high-density microelectrode array (HDMEA), featuring 4096 electrodes in a surface of a few square mm. The acquisition system featured in the setup, the BioCam X from 3·Brain, is capable of recording in parallel from the whole array, with a sampling frequency of 18 kHz per electrode. The electrode density and the elevated sampling frequency allow to analyze the neural activity of a large neuronal assembly at a cellular level with sub-millisecond resolution. Both electrical and optical stimulation is embedded. Electrical stimuli with custom waveform are provided by a PlexStim™ Electrical Stimulator System connected to 16 stimulating electrodes. In this case, the total closed-loop latency from signal acquisition to stimulation stays under 2 ms. For the optical stimulation, a Texas Instruments DLP® LightCrafter™ Evaluation Module is embedded. A constant flow of 608x684 pixels images is projected to the cells under study with a refresh rate of 60 Hz. The total latency to change a full frame, as a response of a neural activity trigger, is 21 ms.

- All functionalities of the processing system have been assessed with *ex vivo* experiments targeting mouse retinas. The system correctly processes all the incoming signal in real-time, generating the required stimuli with the previously reported latencies.

- A biologically relevant experiment has been performed with the developed setup. The aim was to automatically classify the retinal ganglion cells (RGCs) based on their response to different optical stimuli. The implemented protocol was able to cluster the activity of RGCs based on the sub-millisecond correlated (SMC) spikes detected in neighboring channels. Of all the clusters, 90.8±4.5% were classified as ON- or OFF-type cells (standard error mean based on 5 retinas). The others were correctly excluded from classification since their behavior was either ambiguous or elicited a poor neural response to the optical stimuli provided.

- A denoising system has been designed to improve the signal-to-noise ratio (SNR) of the biological signals acquired with HDMEA-based platforms. Exploiting oversampling techniques, the design aims at the reduction of the overall input thermal noise at the cost of part of the input sampling frequency. A prototype, implemented on a Intel® Arria® 10 FPGA, is able to process in real-time all the signals recorded by the BioCam X and it is currently under test.

## 7.2 Future Work

As of now, the closed-loop setup is already being used for neuroscience experiments. The high level of scalability and adaptivity, together with the possibility to easily modify the software for the closed-loop algorithm, render the system a very powerful tool. Nevertheless, many improvements can still be made.

An important feature missing is to provide an easy way to create and validate novel closed-loop algorithms. In the last year, many improvements to render the system more user-friendly were made; software libraries were created to control the hardware modules from the ARM processor with high-level functions. Further work must be made in this direction to ensure that the use of the system is straightforward even for neuroscientists with no background in electronics.

Another important step that should be persuaded is the integration of the denoising system with the closed-loop one. As of now, they are implemented in two different devices. Merging the two will result in a single processing system capable of denoising and performing the subsequent elaboration up until the closed-loop in real-time with minimal latency. Thus improving, even more, the potentiality of the setup as a tool for neuroscience.

Finally, it is desirable for the system to be used in a wide range of closed-loop experiments to tackle the fundamental questions of neurophysiology.

## 7.3 List of Publications

- Seu, G. P., Angotzi, G. N., Tuveri, G., Raffo, L., Berdondini, L., Maccione, A., and Meloni, P. (2017b). On-fpga real-time processing of biological signals from high-density meas: a design space exploration. In *Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2017 IEEE International*, pages 175–183. IEEE
- Seu, G. P., Angotzi, G. N., Tuveri, G., Raffo, L., Berdondini, L., Maccione, A., and Meloni, P. (2017a). A closed-loop system for neural networks analysis through high density meas. In *Ph. D. Research in Microelectronics and Electronics (PRIME), 2017 13th Conference on*, pages 321–324. IEEE
- Seu, G. P., Angotzi, G. N., Boi, F., Raffo, L., Berdondini, L., and Meloni, P. (2018). Exploiting all programmable socs in neural signal analysis: A closed-loop control for large-scale cmos multielectrode arrays. *IEEE Transactions on Biomedical Circuits and Systems*
- Zaher, S., Lonardoni, D., Boi, F., Seu, G. P., Angotzi, G. N., Meloni, P., and Berdondini, L. (2019). A closed-loop system processing high-density electrical recordings and visual stimuli to study retinal circuits properties. In *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)*. IEEE. [Accepted; To Be Published]

# Appendix A

# FPGAs and APSoCs

*"Ah, yes, ah, yes, my dear friend, think it over well: a minute ago, when this thing happened to you, you were a different person; not only that, you were at the same time a hundred others, a hundred-thousand. And believe me, there is no occasion for wonderment in this fact. Look rather and see if it now seems to you so certain that tomorrow you will be what you assume you are today. My dear friend, the truth is this: they are all fixations. Today, you fix yourself in one fashion, tomorrow in another."*
Translated from Uno, nessuno e centomila by Luigi Pirandello
(dramatist, novelist and poet, Nobel Prize in Literature)

## A.1 Brief history of Programmable Logic Devices

The first programmable logic device (PLD) was developed in 1956. Sadly enough, like many other great inventions, it was created for war purpose. We are in the middle of the Cold War, and the United States Air Force wanted a more versatile and safe way to store the constants required for the targeting system of their intercontinental ballistic missiles. The programmable read-only memory (PROM) was realized to satisfy these requirements. Called also field programmable read-only memory (FPROM), the device can be programmed with the required data, after production, directly "in the field" where you need it, differently from the standard read-only memories (ROMs), where data is written during the manufacturing process. For obvious reasons, this technology was held secret for many years and became commercially available only in 1969.
Before explaining how the PROMs works, we have to introduce a new notation for the logic

ports and connections. On the left of fig. A.1, it is depicted the conventional notation for an AND gate with 3 inputs. The new notation that we are going to use for the same port is

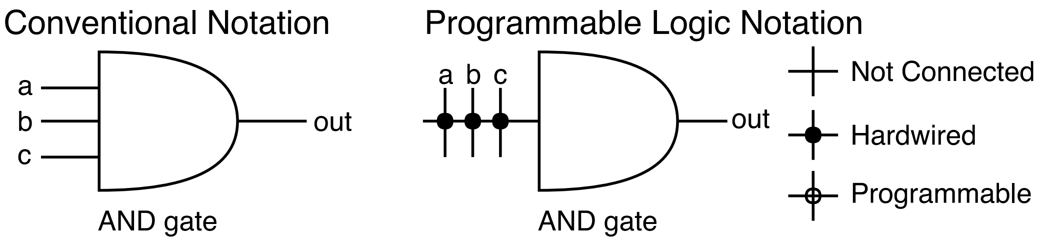


Figure A.1 Conventional notation vs Programmable Logic notation.

depicted on the right part of the same figure. When two wires are crossed, there are three possible situations:

- Not Connected: there is no relation between the two wires.
- Hardwired: the two wires are directly connected; there is a short-circuit between them created during the manufacturing process.
- Programmable: a direct connection can be created between the two wires in the field.

Now back to the PROM, an example of its architecture is depicted in fig. A.2. As we can see, the architecture is composed of two main parts: an AND plane and an OR plane. The AND plane is created with hardwired connections and works as an address decoder. The inputs *a, b, c* are used to select the output word. For example, if $a$=$b$=$c$=‘0’ we are accessing the first word, indeed only the AND gate at the top will have an output ‘1’ that can trigger the OR plane. With $N$ inputs, $2^N$ AND ports are required, and the same amount of words are accessible. The OR plane is instead the storage location, the size of the words stored is equal to the number of outputs. To store a word, we have to program the connections in the OR plane to produce as output the necessary value. For example, if we want to store the value 6 in the first location of the memory, we have to change the connections in the first row of the OR plane in order to obtain the output ‘0110’, so the second and the third programmable connections must become hardwired. With $M$ outputs the same number of OR ports are required, and the words stored are $M$-bits wide. It is easy to see that the architecture in the picture represents an 8x4 bit memory. The working principle behind the PROM is based on fuses or antifuses. The memory is produced holding only ‘1’ bits. To change a bit of the memory to ‘0’ the relative fuse is burned, and the new value is held forever; this is a significant limitation of the PROM. After the data is written in the chip, it cannot be changed

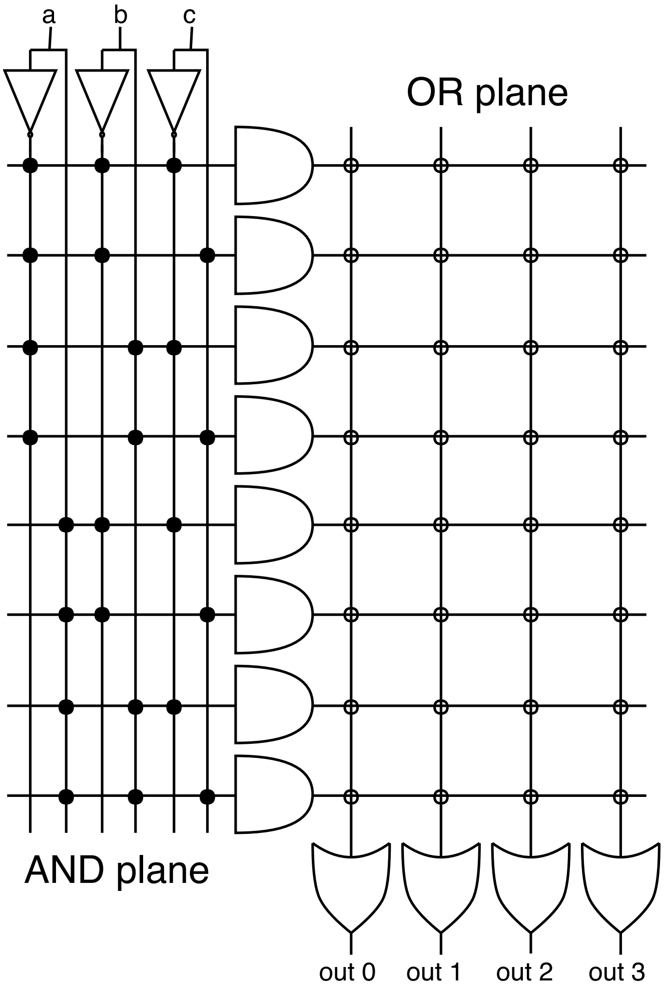


Figure A.2 PROM architecture.

in any way. To overcome this limitation, in 1971, the Erasable Programmable Read-Only Memory (EPROM) has been developed based on a completely different working principle. The EPROM is an array of floating-gate field-effect transistors (FETs). The difference from normal FETs is the presence of an additional floating gate as depicted in Fig. A.3. Floating

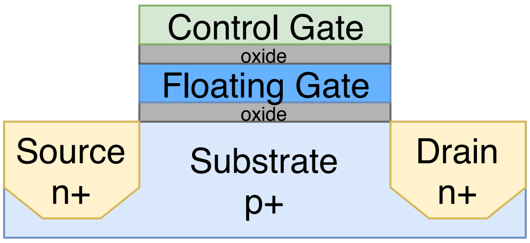
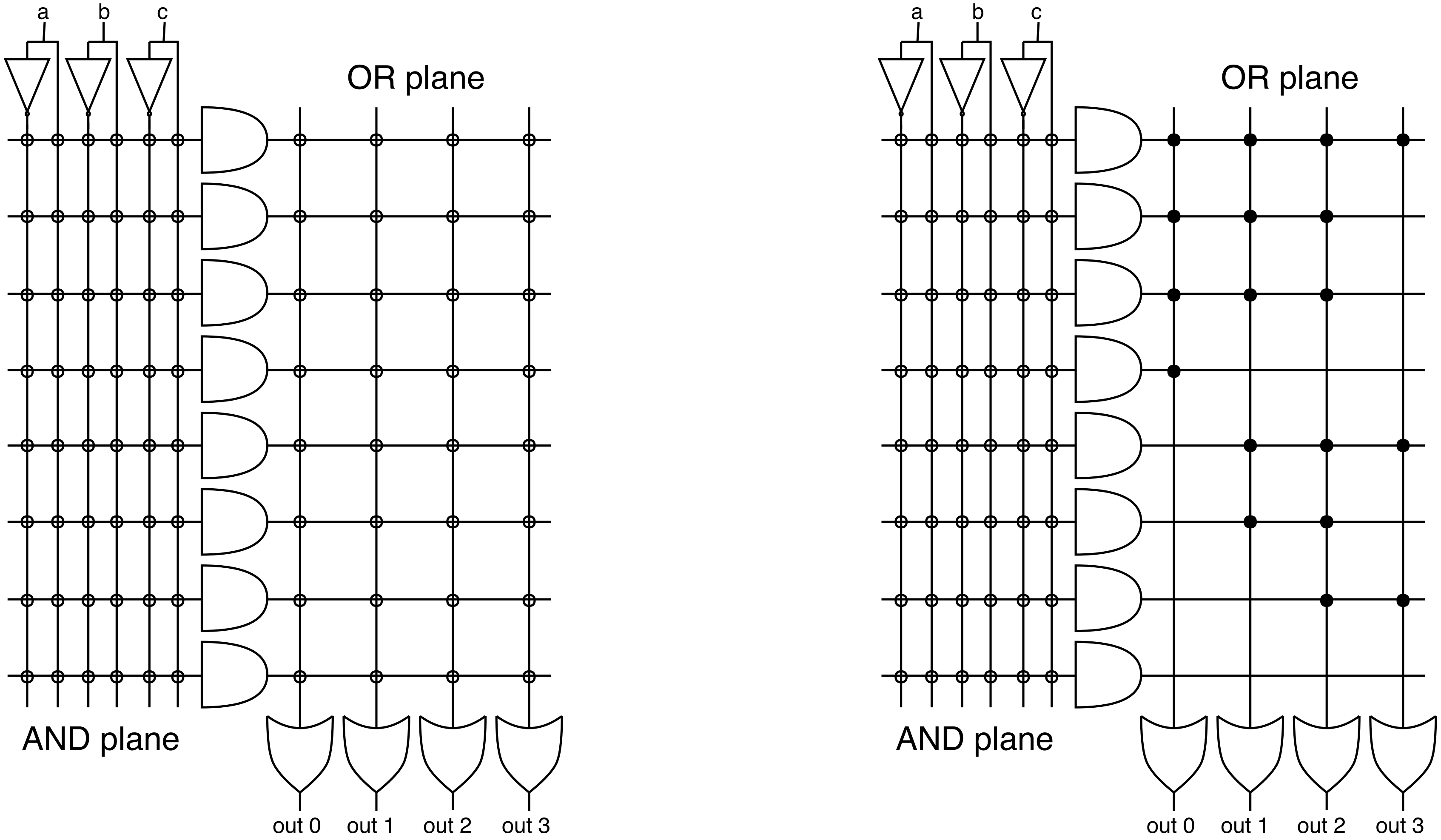


Figure A.3 Floating-gate field-effect transistor.

means that the gate is electrically isolated from all the other parts of the circuit, and only a capacitive coupling exist between it and the control gate. When a high voltage is applied to the control gate, many electrons from the substrate are able to break the energy barrier of the insulator and accumulate in the floating gate. Once the voltage is removed, the electrons are trapped in the floating gate, changing the state of the transistor. In order to remove the charge and take the transistor back to its original state, an ultraviolet light must be applied on the chip. The light ionizes the oxide allowing the trapped electrons to flow away. The erase process is not very practical; the EPROM must be taken out of the circuit and exposed to the UV light, which erases all the memory. To provide a more practical way to erase the contents from the device the electrically erasable programmable read-only memory (EEPROM or E$^2$PROM) was developed in the same year. The EEPROM is very similar to the EPROM, it is composed by an array of floating-gate transistors with a thin layer of dioxide between the floating gate and the substrate, that allows electrons to pass through by the process of quantum tunneling. In 1975, another significant step in the PLD field was achieved with the realization of the programmable logic array (PLA). The difference between the PLA and the PROM is that in the former both the OR and the AND planes are programmable. With this new level of flexibility, the PLAs are ideal to implement logic circuits. The concept exploited here is that many logic functions can be reduced in the form of the sum (OR gates) of products terms (AND gates). Note that also the PROM can be used to create logical functions, but with a much higher resource consumption due to its lack of flexibility in the AND plane, thus unfeasible in many practical situations. Later on, in 1978, the programmable array logic (PAL) became commercially available. The main difference in the PAL is that the OR plane is fixed and the AND plane is programmable. This architecture is quite efficient to develop large logic function, because generally they contain a limited number of product terms. The ability to use all product terms for all the outputs in most cases is useless. By fixing the OR plane, the PAL devices can achieve a much higher working speed. An example of the schematic relative to the PLA and PAL is reported in fig. A.4.
To summarize:

- PROM: have the AND plane fixed, the OR plane is programmable.
- PLA: have both the OR and the AND planes programmable.
- PAL: have the OR plane fixed, the AND plane is programmable.

The evolution of the PAL led to the development of more complex devices containing multiple logic planes and also logic macrocells for specific logic functions. All these components



(a) PLA architecture.

(b) PAL architecture.

Figure A.4 PLA and PAL architecture. The main difference between these architectures is that in the PAL the OR plane is hardwired, while in the PLA is programmable.

embedded in the same chip are called complex programmable logic device (CPLD). Moreover, CPLDs were developed with complementary metal–oxide–semiconductor (CMOS) technology making them reprogrammable, in contrast with most of PLAs and PALs that were programmable only one time. The main company to produce CPLD is Altera, founded in 1983, and as of today one of the two leading companies in the programmable logic devices field. The other leader company is Xilinx, founded in 1984, and responsible for the invention of the field programmable gate arrays (FPGAs) that introduced in the market in 1985. For many years Altera pleaded the superiority of CPLDs, but in the end surrendered to the superiority of FPGAs, and started producing them in 1992.

## A.2 FPGAs

So, we have finally reached the FPGA, the ultimate programmable logic device. As the FPROM, the device is field programmable, meaning that it can be programmed where it is needed: *in the field*. Developed to imitate gate arrays in a more flexible way, FPGAs reached a success that was not even foreseen by its inventors. A gate array is an integrated circuit where the logic functionality is achieved by interconnecting with metal wires the existing transistors that were deployed in the chip. The FPGA has the same principle, but the interconnections are reprogrammable. The device is composed of three main components:

- Logic gates: such as AND, OR, ... used to create complex logic functions.
- Registers: implemented with Flip-Flops and used to store variables.
- Wires: that can be programmed to connect gates and registers.

FPGAs are produced with different technology process, splitting the devices into three categories:

- SRAM-based: static memory cells control the connections. They can be reprogrammed multiple times. When the energy supply is off, they lose the programming; they need to be programmed upon startup every time. Usually, an external memory is used to hold the design after power off.
- Antifuse-based: antifuse CMOS technology control the connections. Can be programmed only the first time, then the design is held forever even without energy supply.

- Flash-based FPGAs: floating gate cells control the connections. A smaller area is used compared to SRAM-based devices. They can be erased and reprogrammed but not infinite times. They hold the design after power off.

One of the main ideas behind FPGAs that made them so powerful is that the logic gates are implemented using lookup tables (LUTs). A LUT is merely a memory array that contains the results of a logic function for all the possible inputs, one or more multiplexers driven by the inputs of the function select the appropriate output. The schematic of a generic 2-input LUT is depicted on the left of fig. A.5. In the figure *a* and *b* are the inputs while *y* is the output. Any 2-input logic port (AND, OR, NAND, ...) can be implemented with such architecture by simply storing the appropriate values. For example, on the right of the fig. A.5 is an implementation of a 2-input AND gate using the LUT. As it is easy to notice only when both *a* and *b* are equal to '1' then the output *y* is '1'. The use of LUTs gives many great advantages. The results of the logic function are stored once and never calculated again; this can speed up significantly the processing since reading the value from memory is usually done much faster than calculating it again. Moreover, an architecture made of LUT has excellent scalability, merging a couple of 2-input LUTs we can create a 3-input LUT, capable of implementing any 3-input logic function. In this way, any complex logic function can be implemented with limited hardware resources, differently from the CPLDs that do not scale so well.

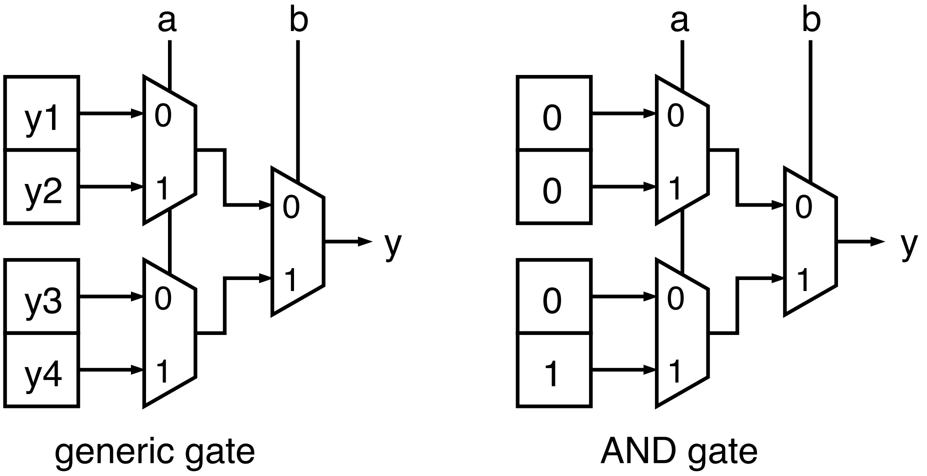


Figure A.5 On the left the schematic of a generic 2-input lookup table, on the right the implementation of an AND gate.

Usually, in the FPGAs, one or more LUTs with one or more Flip-Flops are merged in macro-blocks called configurable logic blocks (CLBs). The CLBs also contain additional hardware to optimize the creation of specific functions. For example in the 7 Series FPGAs from Xilinx each CLB contains:

- 2 LUT5 (5-input LUT) that can be merged to create a LUT6.
- 2 Flip-Flops to store the output of the LUTs.
- A dedicated carry logic for arithmetic functions.
- Wide multiplexers.

The typical FPGA architecture that results from this is depicted in fig. A.6. It is composite of an array of CLBs interconnected by wires and routers while a series of I/O buffers provide connectivity with the outside. By programming the CLBs and the routers we can implement any possible logic circuit, but those familiar with digital logic circuits soon notice that, by its nature, an FPGA is less efficient than an application-specific integrated circuit (ASIC) that can be used for the same purpose. An ASIC system is designed similarly to an FPGA one. The functionality is described with a hardware description language (HDL), the most used being Verilog and VHDL. The difference is that the ASIC is realized fabricating the specific chip with the desired functionality, while the FPGA device, already fabricated, is

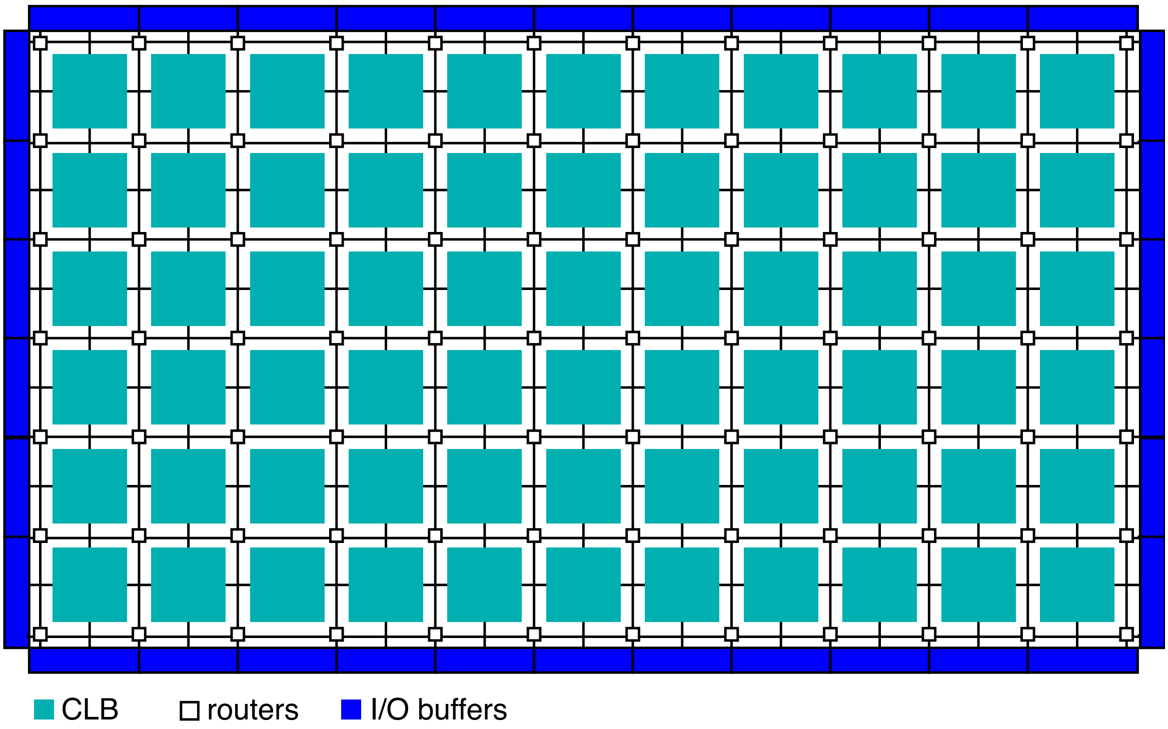


Figure A.6 Example of a FPGA architecture

merely programmed. Looking back at fig. A.5, it is easy to notice that an AND or OR gate implemented with a LUT is for sure bigger and slower than its hardwired realization in the integrated circuit (IC). Moreover, the wires going through the routers (fig. A.6) to connect multiple CLBs altogether are again bigger and slower than the direct wires connecting the gates in an ASIC.

Then what is the FPGA used for if the ASIC is always more efficient?

Three fields to use the FPGA can be identified. First, for prototyping a circuit that will then be implemented on an ASIC. An FPGA can be programmed all the times that we want, this feature makes it great for testing the functionality of a design before its fabrication. If a bug is found in the design while testing it on the FPGA, it can be immediately corrected and retested on the same FPGA, instead, a bug found on an ASIC after production is extremely expensive. To fabricate an ASIC the masks relative to the required design must be produced first, with a very high cost in terms of money but also time. So FPGA prototyping is highly recommended before implementing the design on an ASIC to avoid the production of chips presenting bugs. We have just said how expensive and time consuming is to produce an ASIC, this introduces us to the second and third field where FPGAs must be preferred to ASICs. In a world evolving incredibly fast, sometimes it is better to have a less efficient product ready immediately, rather than to wait months to have the most efficient device possible. When the time-to-market is crucial, a late release of a product will result in commercial failure; this is the second field where FPGA is significantly better. Finally, for products with a small market, it is not possible to afford the development cost of an ASIC, it would be impossible to recover the initial investment. FPGAs, again, are to be preferred to ASIC implementations.

Summarizing: FPGAs are used for prototyping or for low-volume products for which costs in terms of time and money are strictly constrained, in any other cases (the vast majority) ASICs are to be preferred.

**Wrong!** In the following chapter we will see why FPGAs are much more used than this.

## A.3 APSoCs

The first empirical evidence that FPGAs are not only confined to low-volume products is the acquisition of Altera by Intel in 2015 for $16.7 billion, not 16.7 million, not 16.7 thousand, not 16.7 hundred, not 16.7, not 16.7 cents but $16.7 billion!

Back to theory, first we must stress that FPGAs are programmable, ASICs are not. To have programmability in an FPGA we lose a bit of performance and efficiency, but programmability is a great feature. Most of the processing architectures are programmable: central processing units (CPUs), digital signal processors (DSPs) and graphical processing units (GPUs). We have to compare FPGAs to them not (only) with ASICs. CPUs, DSPs, and GPUs are processors optimized for a specific function. CPUs are used for general purpose application, where the variability of possible instructions to execute is higher. DSPs are instead optimized for digital signal processing, i.e., filters with many multiplications. Finally, GPUs are optimized to manipulate images, blocks of data composite of many pixels, thus for the execution of parallel instructions on different data. But can a field programmable gate array compete in any of this specific field against dedicated hardware? Well no, that is why FPGA evolved into something much more powerful and versatile that we will call All Programmable System on Chip (APSoC). A System on Chip (SoC) is an integrated circuit with many different interconnected electronic components on it. Usually contains at least a CPU, a memory and some input/output ports, but more digital or analog components can be included and also radio-frequency functions. All this in a single chip, making it smaller, more power efficient and usually cheaper than alternatives. An All Programmable SoC is something similar, a chip with a bunch of different components, that can be programmed to implement the desired design. The term APSoC, as far as I know, was originally used by Xilinx for a particular kind of FPGAs that started producing in 2011. Later on, the name was abandoned and substituted by Zynq®-7000 SoC.

> **Zynq®-7000 SoC**
> (Xilinx, 2018): The Zynq®-7000 SoC family integrates the software programmability of an ARM®-based processor with the hardware programmability of an FPGA, enabling key analytics and hardware acceleration while integrating CPU, DSP, ASSP, and mixed signal functionality on a single device. Consisting of single-core Zynq-7000S and dual-core Zynq-7000 devices, the Zynq-7000 family is the best price to performance-per-watt, fully scalable SoC platform for your unique application requirements.

In 2012 Altera (now Intel) started producing similar devices and they call them SoC FPGAs.

> **Intel® SoC FPGA**
> (Intel, 2018): Intel® SoC FPGAs integrate an ARM®-based hard processor system (HPS) consisting of processor, peripherals, and memory interfaces with the FPGA fabric using a high-bandwidth interconnect backbone. It combines the performance and power savings of hard intellectual property (IP) with the flexibility of programmable logic. These devices include additional hard logic such as PCI Express Gen2 and Gen3, multiport memory controllers, error cor-

> rection code (ECC), memory protection and high-speed serial transceivers. The ARM-compatible software provides unmatched target visibility, control, and productivity using our FPGA-adaptive debugging.

Also Microsemi, another minor FPGA producer, call its devices with similar architecture SoC FPGAs. In the following, to avoid confusion, the original term *All Programmable System on Chip* will be used when speaking in general of this kind of devices, while the other two terms *Zynq*$^{®}$*-7000 SoC* and *Intel*$^{®}$ *SoC FPGAs* will be used for the specific company made devices. To easy understand what an APSoC is, and how different is from a standard (old) FPGA we can take a look at fig. A.7, where a simplified version of its architecture is depicted. As we can notice, 3 new components are present with respect to the pure FPGA on fig. A.6: CPU, DSP and BRAM. These new components are hardware blocks optimized for specific functions, and cannot be configured as the CLBs. This does not limit too much the possibilities, instead allows the APSoC to achieve excellent performance also in other fields. With this new architecture, the CLBs are mainly used as glue logic to connect all the other hardware blocks in order to achieve the required functionality. The hard processor gives the system more degree of programmability, allowing the execution of software code (easy to write and modify), ideal for control tasks. The DSP slices, instead, are optimized for

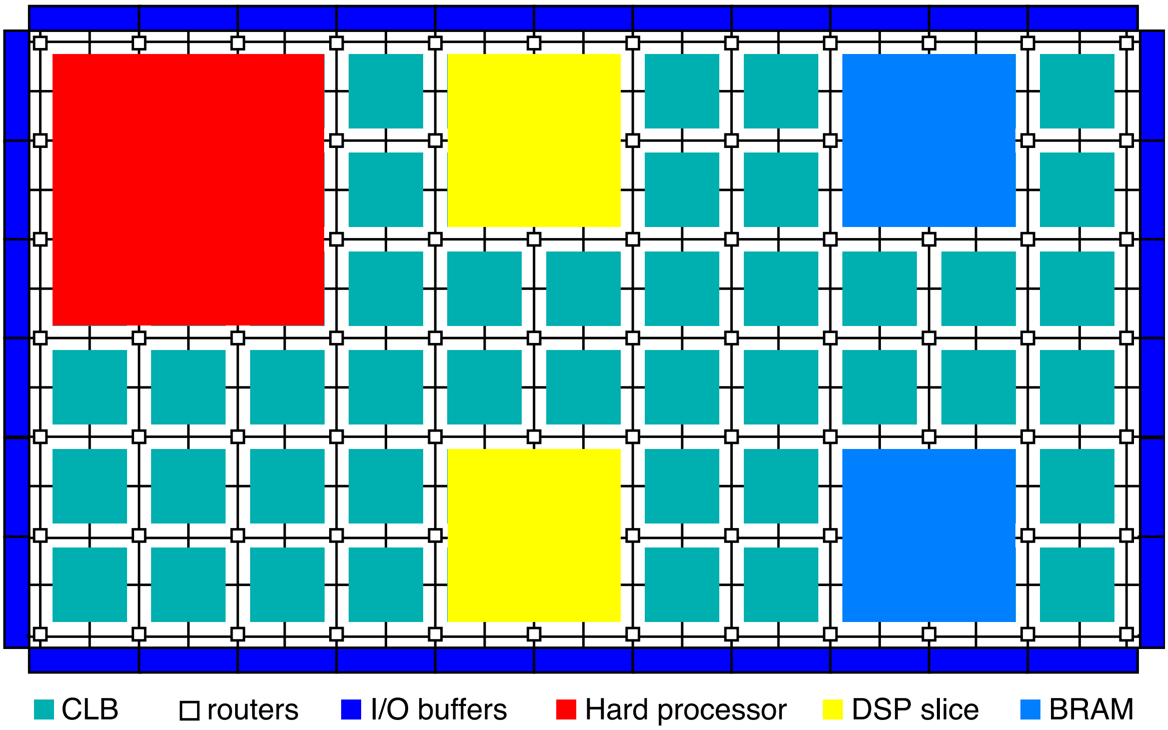


Figure A.7 Example of a All Programmable System on Chip architecture

multiply-accumulate (MAC) operations, precisely what you need for digital filters and signal processing in general. Finally, BRAM stands for block RAM (random access memory). Each block is a dedicated memory, containing much more memory than what you can store using LUTs, and can be used to store partial results directly inside the FPGA. Since the time to read/write data inside these blocks from the FPGA is practically zero, a significant speedup can be achieved compared to storing and reading the data to and from an external memory. The fig. A.7 is not exactly in scale, many more DSP slices and BRAMs are present in a single chip, with the high-end devices holding even thousands of such blocks, schematics of real devices will be presented in section A.4. With these new upgrades the FPGAs, now APSoCs, are not only able to compete with CPUs and DSPs, but greatly outperform them in most applications in terms of efficiency and performance. Why then CPUs and DSPs have not disappeared yet? Because performance and efficiency are not the only parameters that matter. For example, APSoCs are programmable but not easy to program. To use an APSoC the skills and competences required are beyond most of the basic programmers. Even with the right experience and knowledge, designing a system for an APSoC requires much more time and effort than writing software for a CPU or a DSP. So it is not easy to say that an architecture is always better than another one. As a rule of thumb: if a particular device is still in production, it means that it is useful for some application. In fig. A.7 we have seen the hard processor and the DSP slices that make the APSoC more performing than CPUs and DSPs, what about GPUs? FPGAs do not have "GPU slices" and cannot compete with GPU in graphical applications, in other fields, instead, the challenge is on. A simple "FPGA vs GPU" search on Google brings thousands of articles/forums/question-and-answer websites dealing with the subject. Machine learning applications are one hot topic where the fight between FPGAs (APSoC) and GPUs is going on. GPUs have the advantage of being very easy to program exploiting their performances even from software developers with little understanding of the underlying hardware. But especially in the sub-field of deep neural networks, FPGAs take advantage of their better flexibility and exploiting the availability of distributed on-chip memory (BRAMs) and the incredible raw computational power (DSP slices) they seem to become the winners of this competition achieving better performances and power efficiency.

One last thing about FPGAs, recently some of them come with another great feature: dynamic partial reconfigurability. This allows reprogramming part of the device while the rest is still running. Basically, a controller, that can be inside or outside the FPGA, is in charge of loading the required design in the device, and changing parts of it as needed by the application, while the remaining of the device keeps running and executing its own tasks. A wide range of new applications are available for these new devices, and the old ones can significantly benefit from this new feature. The same chip can now be shared among many tasks using a time division multiplexing strategy, bringing two main benefits: cost reduction, indeed a smaller chip can achieve the same functionality of a bigger one; better performances, thanks to the ability to adapt and optimize the system to the current data being processed loading specific hardware.

## A.4 Examples of commercial APSoCs

In this last section are reported images from real devices to bring more reality to the theory presented before. The images are extracted from the software used to design and implement the projects in the FPGA devices.

### A.4.1 Xilinx

We start with the Xilinx products. The software used is Vivado Design Suite, the target board is the Zynq-7020 SoC. In fig. A.8 we can see a little part of the whole device where are highlighted the various blocks. On the left in yellow two DSP slices, in the middle an array of CLBs and in the right 2 BRAMs able to hold 18 kbits each. As suggested by the grey shape behind the BRAMs, they can be merged to form a bigger memory with a total capacity of 36 kbits. In the top-left CLB are highlighted his subcomponents, 4 LUT6 (6-input LUT), 3 multiplexers, a carry logic for arithmetic functions and finally 8 flip-flops to register the results. Not necessarily all these subcomponents are used at the same time, more likely, for a given design, only a subpart of the available resources will be used. This is clear in the next figure A.9, here a broader view of the same device is depicted, but, this time, only the resources used for a particular design are highlighted. The main purpose of this picture is to show how highly interconnected are the various component, the white arrows show the direct connection (input and output) between the selected DSP and the other resources. As already said, we can notice that even when a CLB is used, it does not mean that all its subcomponents are used. Another thing that can be noticed is that some BRAMs are used as 18k bit memories, while other are merged together to form 36 kbits memories (the one exactly in the center of the picture and the one on the far right). Finally, in the last picture for the Xilinx products, fig. A.10, is depicted the full schematic of the Zynq-7020 with highlighted the used resources for a specific design. In the top-left corner, there is the ARM-based hard processor that occupy a significant portion of the chip. The remaining of the chip is

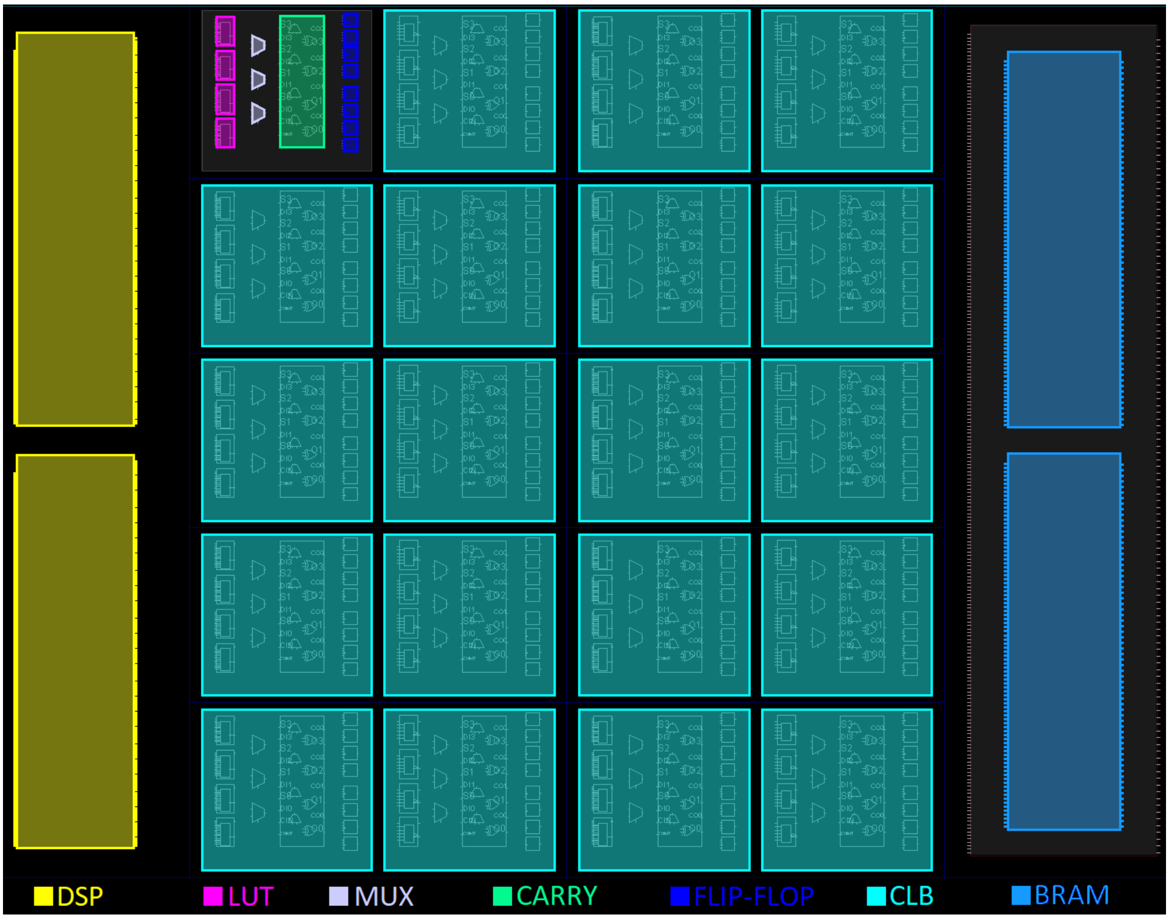


Figure A.8 Zoom in inside a Xilinx Zynq-7020 SoC with highlighted the various components. The CLBs are the classical components of the FPGAs while DSPs and BRAMs have been featured only in recent devices.

composed of columns of DSP and BRAM in a "sea" of CLBs. Another kind of hardware block, highlighted in green, is the phase-locked loop (PLL) that uses a signal generated by an oscillator to create the clock signals for the FPGA with the required frequency. In the far right and far left, with different colors, are the pads to connect the device with the outside world. These pads are used to receive the inputs to elaborate and to send out the generated outputs, but also to connect the APSoC with external memories or other devices.

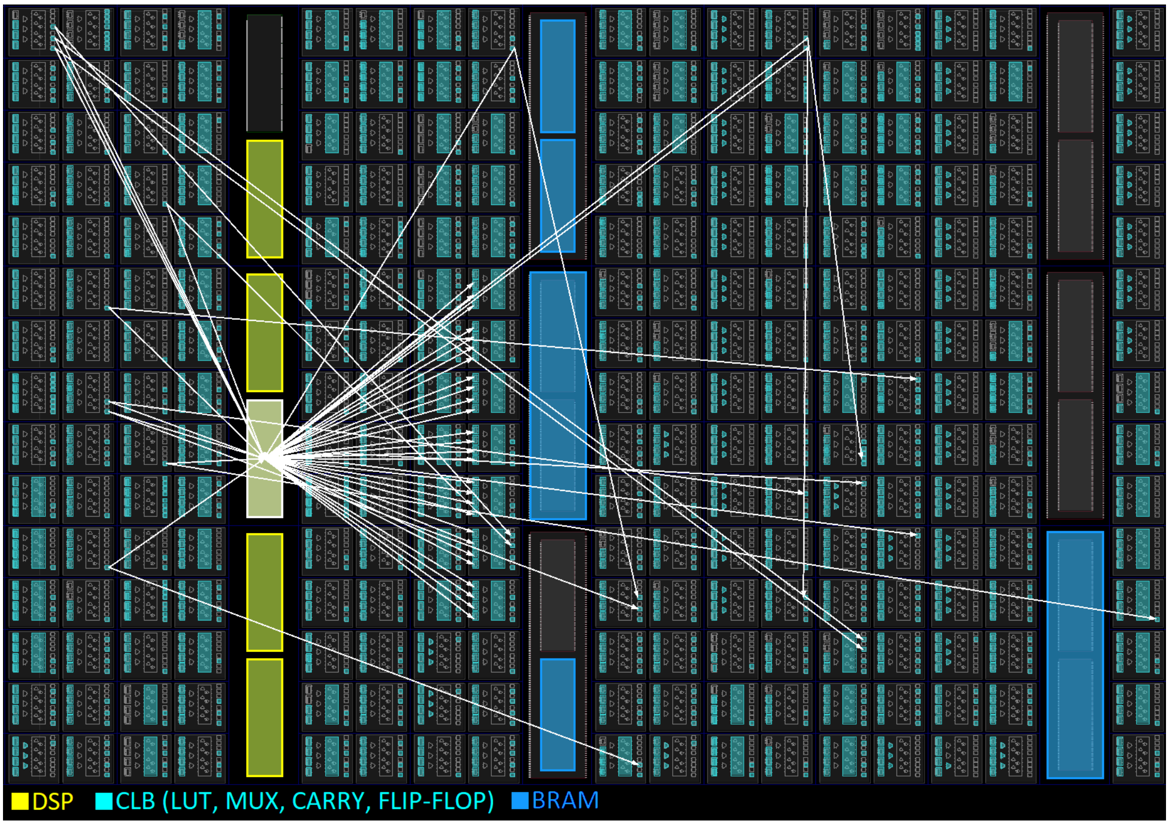


Figure A.9 Zoom in inside a Xilinx Zynq-7020 SoC to show the high degree of interconnection of the various components.

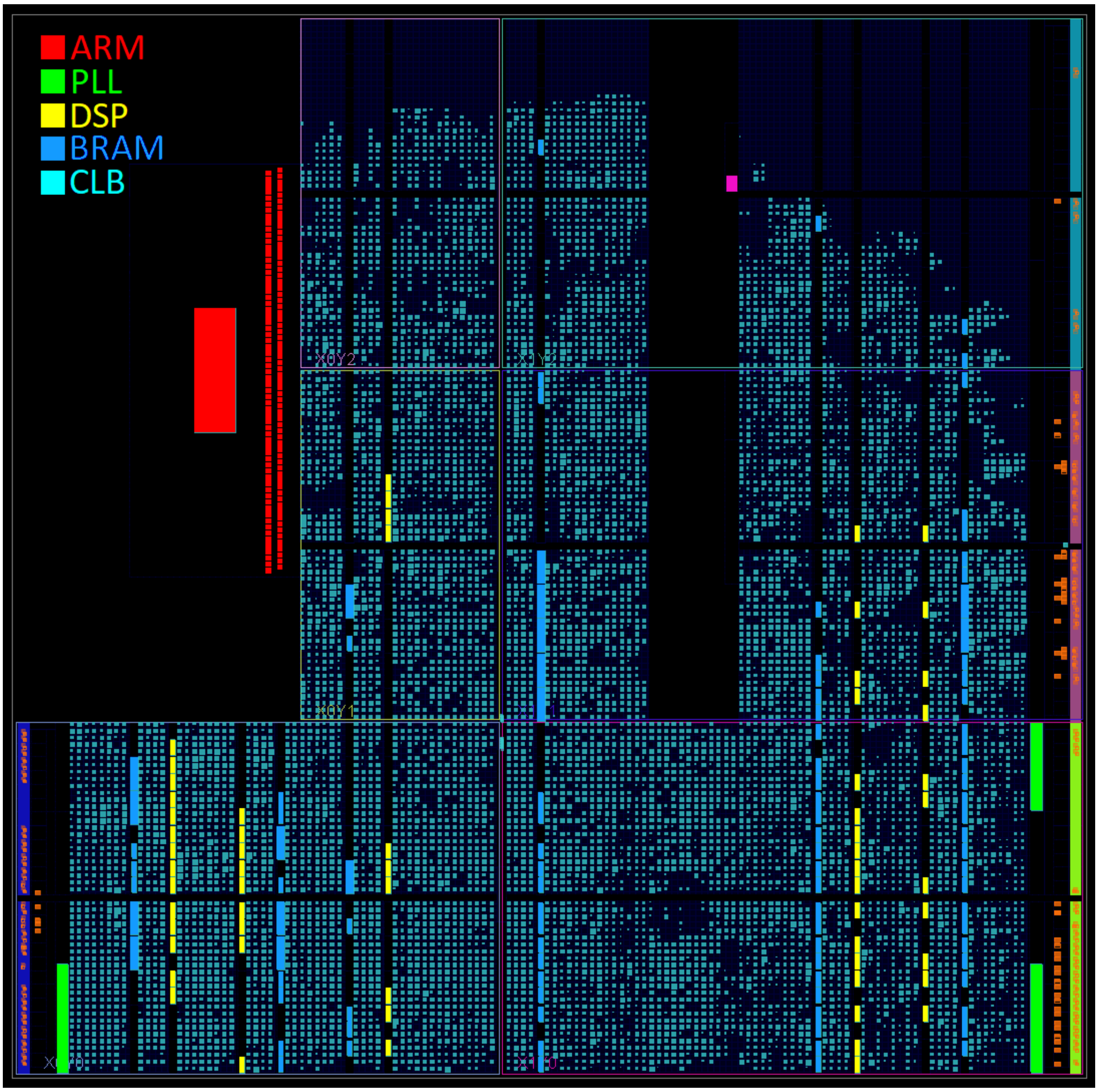


Figure A.10 Full schematic of the Xilinx Zynq-7020 with highlighted the resources used for a particular design. The percentages of used resources are: LUT 54%, FF 27%, BRAM 42%, DSP 28%, PLL 37.5%.

### A.4.2 Intel (formerly Altera)

The software used for designing and implementing systems in the Intel FPGAs is Quartus Prime. From its additional tool Chip Planner is extracted the fig. A.11 that represent the full schematic of the Cyclone V ST SoC FPGA. Here the ARM-based hard processor is in the top-right corner, the remaining of the device is similar to the Zynq-7020 that we have seen before. Columns of DSPs and BRAMs are in an array of CLBs. Of course, the basic blocks from Intel are slightly different from the basic blocks of Xilinx. Moreover, different device families in the same company can have different basic components. For example in the Cyclone V of fig. A.11 each BRAM can hold a maximum of 10k bits, instead, in another Intel device, the Arria 10, each BRAM can hold up to 20k bits.

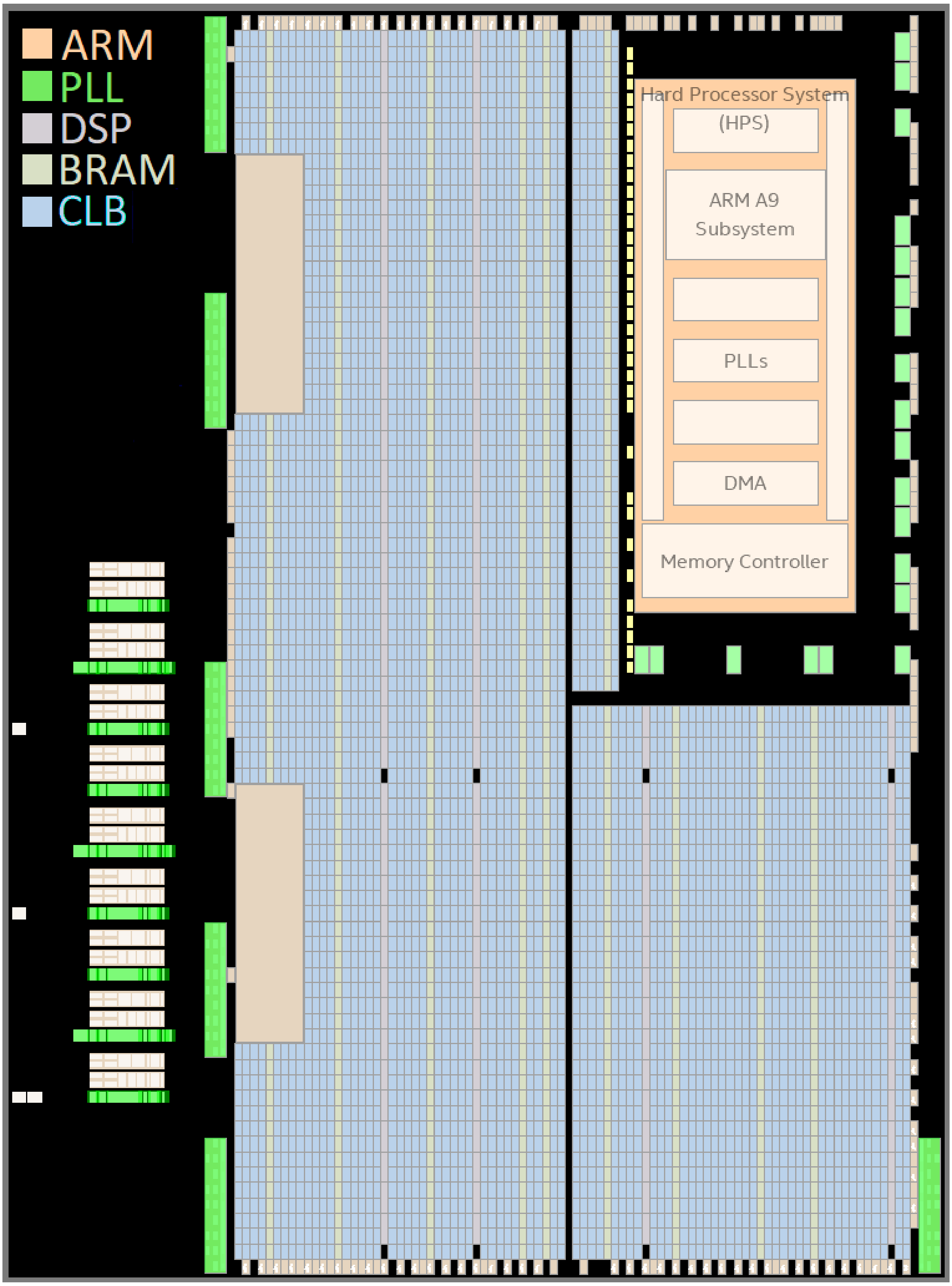


Figure A.11 Full schematic of the Intel Cyclone V ST SoC FPGA.

## A.5 Conclusions

The history of PLD started more than half a century ago, with the first commercial PLD going into market exactly 50 years ago. A summary of the most important steps are reported in fig. A.12. Since their birth in 1985, FPGAs have kept improving, and the fields of application raised as well. As of now, the FPGA market is valued about $6 billion per year, and it is forecast to reach $10 billion by 2023 according to a research report from MarketsandMarkets™. In my humble opinion, FPGAs will keep rising in popularity in the next years, thanks to the incredible performance and unique peculiarities compared to other processing architectures.

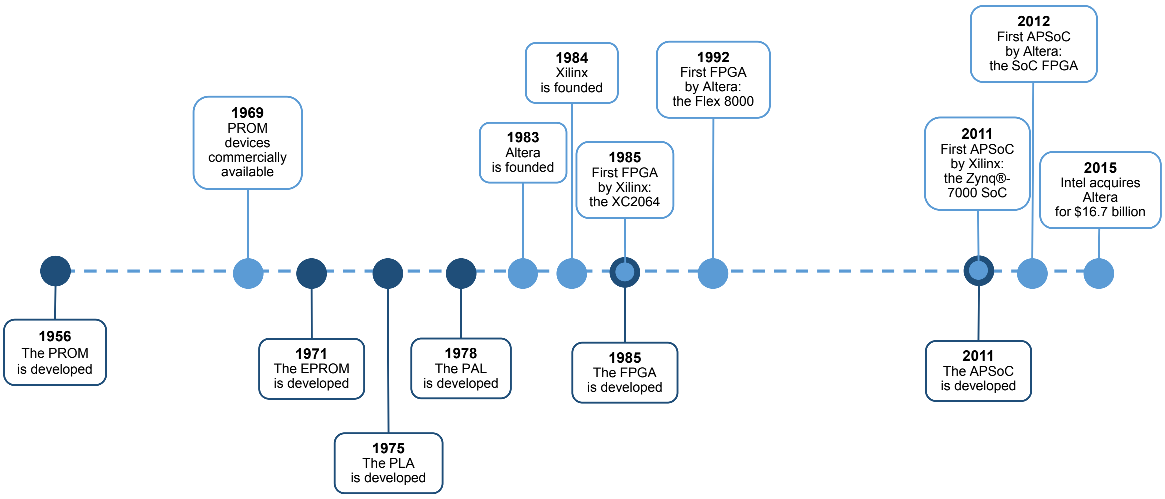

1956 The PROM is developed
1969 PROM devices commercially available
1971 The EPROM is developed
1975 The PLA is developed
1978 The PAL is developed
1983 Altera is founded
1984 Xilinx is founded
1985 First FPGA by Xilinx: the XC2064
1985 The FPGA is developed
1992 First FPGA by Altera: the Flex 8000
2011 First APSoC by Xilinx: the Zynq®-7000 SoC
2011 The APSoC is developed
2012 First APSoC by Altera: the SoC FPGA
2015 Intel acquires Altera for $16.7 billion

Figure A.12 PLD history timeline.

Used sources or suggested to go more into deep: Becker et al. (2007); Guerra (2016); Kreinin (2013); Trimberger (2015).

# References

3·Brain (2018). BioCam X. http://www.3brain.com/biocamx.html. Accessed: 2018-10-31.

Abee, C. R., Mansfield, K., and Tardif, S. D. (2012). *Nonhuman primates in biomedical research: biology and management*, volume 1. Academic Press.

Aldini, G. (1804). *Essai theorique et experimental sur le galvanisme, avec une serie d'experiences faites en presence des commissaires de l'Institut national de France, et en divers amphitheatres anatomiques de Londres, par Jean Aldini... Avec planches*. De l'imprimerie de Fournier Fils.

Angotzi, G. N., Boi, F., Lecomte, A., Miele, E., Malerba, M., Zucca, S., Casile, A., and Berdondini, L. (2018). Sinaps: an implantable active pixel sensor cmos-probe for simultaneous large-scale neural recordings. *Biosensors and Bioelectronics*.

Angotzi, G. N., Boi, F., Zordan, S., Bonfanti, A., and Vato, A. (2014). A programmable closed-loop recording and stimulating wireless system for behaving small laboratory animals. *Scientific reports*, 4:5963.

Azevedo, F. A., Carvalho, L. R., Grinberg, L. T., Farfel, J. M., Ferretti, R. E., Leite, R. E., Filho, W. J., Lent, R., and Herculano-Houzel, S. (2009). Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *Journal of Comparative Neurology*, 513(5):532–541.

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.

Barry, J. R., Lee, E. A., and Messerschmitt, D. G. (2012). *Digital communication*. Springer Science & Business Media.

Becker, J., Hubner, M., Hettich, G., Constapel, R., Eisenmann, J., and Luka, J. (2007). Dynamic and partial fpga exploitation. *Proceedings of the IEEE*, 95(2):438–452.

Berdondini, L., Imfeld, K., Maccione, A., Tedesco, M., Neukom, S., Koudelka-Hep, M., and Martinoia, S. (2009). Active pixel sensor array for high spatio-temporal resolution electrophysiological recordings from single cell to large scale neuronal networks. *Lab on a Chip*, 9(18):2644–2651.

Berlucchi, G. (2017). Neuropsychology. In *Reference Module in Neuroscience and Biobehavioral Psychology*. Elsevier.

Bertotti, G., Velychko, D., Dodel, N., Keil, S., Wolansky, D., Tillak, B., Schreiter, M., Grall, A., Jesinger, P., Röhler, S., et al. (2014). A cmos-based sensor array for in-vitro neural tissue interfacing with 4225 recording sites and 1024 stimulation sites. In *Biomedical Circuits and Systems Conference (BioCAS), 2014 IEEE*, pages 304–307. IEEE.

Bianucci, P. (2008). *Te lo dico con parole tue: la scienza di scrivere per farsi capire*, volume 1. Zanichelli Editore IT.

Biffi, E., Ghezzi, D., Pedrocchi, A., and Ferrigno, G. (2010). Development and validation of a spike detection and classification algorithm aimed at implementation on hardware devices. *Computational intelligence and neuroscience*, 2010:8.

Buzsáki, G., Anastassiou, C. A., and Koch, C. (2012). The origin of extracellular fields and currents—eeg, ecog, lfp and spikes. *Nature reviews neuroscience*, 13(6):407.

Buzsáki, G., Stark, E., Berényi, A., Khodagholy, D., Kipke, D. R., Yoon, E., and Wise, K. D. (2015). Tools for probing local circuits: high-density silicon probes combined with optogenetics. *Neuron*, 86(1):92–105.

Cheung, K. C. (2007). Implantable microscale neural interfaces. *Biomedical microdevices*, 9(6):923–938.

Choi, J. H., Jung, H. K., and Kim, T. (2006). A new action potential detector using the mteo and its effects on spike sorting systems at low signal-to-noise ratios. *IEEE Transactions on Biomedical Engineering*, 53(4):738–746.

Cong, P., Karande, P., Landes, J., Corey, R., Stanslaski, S., Santa, W., Jensen, R., Pape, F., Moran, D., and Denison, T. (2014). A 32-channel modular bi-directional neural interface system with embedded dsp for closed-loop operation. In *European Solid State Circuits Conference (ESSCIRC), ESSCIRC 2014-40th*, pages 99–102. IEEE.

Dalenoort, G. (1982). In search of the conditions for the genesis of cell assemblies: A study in self-organization. *Journal of Social and Biological Structures*, 5(2):161–187.

Enke, C. and Nieman, T. A. (1976). Signal-to-noise ratio enhancement by least-squares polynomial smoothing. *Analytical Chemistry*, 48(8):705A–712A.

Ethier, C., Gallego, J., and Miller, L. E. (2015). Brain-controlled neuromuscular stimulation to drive neural plasticity and functional recovery. *Current opinion in neurobiology*, 33:95–102.

Franke, F., Jäckel, D., Dragas, J., Müller, J., Radivojevic, M., Bakkum, D., and Hierlemann, A. (2012). High-density microelectrode array recordings and real-time spike sorting for closed-loop experiments: an emerging technology to study neural plasticity. *Frontiers in neural circuits*, 6:105.

Frey, U., Egert, U., Heer, F., Hafizovic, S., and Hierlemann, A. (2009). Microelectronic system for high-resolution mapping of extracellular electric fields applied to brain slices. *Biosensors and Bioelectronics*, 24(7):2191–2198.

Frey, U., Sedivy, J., Heer, F., Pedron, R., Ballini, M., Mueller, J., Bakkum, D., Hafizovic, S., Faraci, F. D., Greve, F., et al. (2010). Switch-matrix-based high-density microelectrode array in cmos technology. *IEEE Journal of Solid-State Circuits*, 45(2):467–482.

G, L. B. and Kriszta, V. (1985). Process control: Instrument engineers handbook.

Galvani, L. (1791). De viribus electricitatis in motu musculari: Commentarius. *Bologna: Tip. Istituto delle Scienze, 1791; 58 p.: 4 tavv. ft; in 4.; DCC. f. 70.*

Gibson, S., Judy, J. W., and Marković, D. (2012). Spike sorting: The first step in decoding the brain: The first step in decoding the brain. *IEEE Signal processing magazine*, 29(1):124–143.

Grosenick, L., Marshel, J. H., and Deisseroth, K. (2015). Closed-loop and activity-guided optogenetic control. *Neuron*, 86(1):106–139.

Guerra, M. (2016). The principles of fpgas. [Online; posted 20-May-2016; Accessed: 30-October-2018].

Hafizovic, S., Heer, F., Ugniwenko, T., Frey, U., Blau, A., Ziegler, C., and Hierlemann, A. (2007). A cmos-based microelectrode array for interaction with neuronal cultures. *Journal of neuroscience methods*, 164(1):93–106.

Harris, J. J., Jolivet, R., and Attwell, D. (2012). Synaptic energy use and supply. *Neuron*, 75(5):762–777.

Hebb, D. O. (1963). *The Organizations of Behavior: a Neuropsychological Theory*. Lawrence Erlbaum.

Hierlemann, A., Muller, J., Bakkum, D., and Franke, F. (2015). Highly integrated cmos microsystems to interface with neurons at subcellular resolution. In *Electron Devices Meeting (IEDM), 2015 IEEE International*, pages 13–2. IEEE.

Hilgen, G., Pirmoradian, S., Pamplona, D., Kornprobst, P., Cessac, B., Hennig, M. H., and Sernagor, E. (2017a). Pan-retinal characterisation of light responses from ganglion cells in the developing mouse retina. *Scientific reports*, 7:42330.

Hilgen, G., Sorbaro, M., Pirmoradian, S., Muthmann, J.-O., Kepiro, I. E., Ullo, S., Ramirez, C. J., Encinas, A. P., Maccione, A., Berdondini, L., et al. (2017b). Unsupervised spike sorting for large-scale, high-density multielectrode arrays. *Cell reports*, 18(10):2521–2532.

Imfeld, K., Maccione, A., Gandolfo, M., Martinoia, S., Farine, P.-A., Koudelka-Hep, M., and Berdondini, L. (2009). Real-time signal processing for high-density microelectrode array systems. *International Journal of Adaptive Control and Signal Processing*, 23(11):983–998.

Imfeld, K., Neukom, S., Maccione, A., Bornat, Y., Martinoia, S., Farine, P.-A., Koudelka-Hep, M., and Berdondini, L. (2008). Large-scale, high-resolution data acquisition system for extracellular recording of electrophysiological activity. *IEEE Transactions on biomedical engineering*, 55(8):2064–2073.

Intel (2018). Intel® soc fpga. https://www.intel.com/content/www/us/en/products/programmable/soc.html. Accessed: 2018-10-31.

Islam, M. K., Rastegarnia, A., Nguyen, A. T., and Yang, Z. (2014). Artifact characterization and removal for in vivo neural recording. *Journal of neuroscience methods*, 226:110–123.

Kandel, E. R., Schwartz, J. H., Jessell, T. M., of Biochemistry, D., Jessell, M. B. T., Siegelbaum, S., and Hudspeth, A. (2000). *Principles of neural science*, volume 4. McGraw-hill New York.

Kassiri, H., Tonekaboni, S., Salam, M. T., Soltani, N., Abdelhalim, K., Velazquez, J. L. P., and Genov, R. (2017). Closed-loop neurostimulators: a survey and a seizure-predicting design example for intractable epilepsy treatment. *IEEE transactions on biomedical circuits and systems*, 11(5):1026–1040.

Kim, K. H. and Kim, S. J. (2000). Neural spike sorting under nearly 0-db signal-to-noise ratio using nonlinear energy operator and artificial neural-network classifier. *IEEE Transactions on Biomedical Engineering*, 47(10):1406–1411.

Kreinin, Y. (2013). How fpgas work, and why you'll buy one. [Online; posted 20-June-2013; Accessed: 30-October-2018].

Krucoff, M. O., Rahimpour, S., Slutzky, M. W., Edgerton, V. R., and Turner, D. A. (2016). Enhancing nervous system recovery through neurobiologics, neural interface training, and neurorehabilitation. *Frontiers in neuroscience*, 10:584.

Lin, C.-T., Ko, L.-W., Chang, C.-J., Wang, Y.-T., Chung, C.-H., Yang, F.-S., Duann, J.-R., Jung, T.-P., and Chiou, J.-C. (2009). Wearable and wireless brain-computer interface and its applications. In *International Conference on Foundations of Augmented Cognition*, pages 741–748. Springer.

Ling, G. and Gerard, R. (1949). The normal membrane potential of frog sartorius fibers. *Journal of cellular and comparative physiology*, 34(3):383–396.

Liu, X., Demosthenous, A., and Donaldson, N. (2007). On the noise performance of pt electrodes. In *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, pages 434–436. IEEE.

Liu, X., Lu, Y., Iseri, E., Shi, Y., and Kuzum, D. (2018). A compact closed-loop optogenetics system based on artifact-free transparent graphene electrodes. *Frontiers in neuroscience*, 12:132.

Liu, X., Wan, H., Shang, Z., and Shi, L. (2015). Automatic extracellular spike denoising using wavelet neighbor coefficients and level dependency. *Neurocomputing*, 149:1407–1414.

Liu, X., Zhang, M., Richardson, A. G., Lucas, T. H., and Van der Spiegel, J. (2017). Design of a closed-loop, bidirectional brain machine interface system with energy efficient neural feature extraction and pid control. *IEEE transactions on biomedical circuits and systems*, 11(4):729–742.

Lobel, D. A. and Lee, K. H. (2014). Brain machine interface and limb reanimation technologies: Restoring function after spinal cord injury through development of a bypass system. *Mayo Clinic Proceedings*, 89(5):708 – 714.

Lopez, C. M., Andrei, A., Mitra, S., Welkenhuysen, M., Eberle, W., Bartic, C., Puers, R., Yazicioglu, R. F., and Gielen, G. G. (2014). An implantable 455-active-electrode 52-channel cmos neural probe. *IEEE Journal of Solid-State Circuits*, 49(1):248–261.

Lopez, C. M., Chun, H. S., Berti, L., Wang, S., Putzeys, J., Bulcke, C. V. D., Weijers, J., Firrincieli, A., Reumers, V., Braeken, D., and Helleputte, N. V. (2018a). A 16384-electrode 1024-channel multimodal cmos mea for high-throughput intracellular action potential measurements and impedance spectroscopy in drug-screening applications. In *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, pages 464–466.

Lopez, C. M., Chun, H. S., Wang, S., Berti, L., Putzeys, J., Van Den Bulcke, C., Weijers, J.-W., Firrincieli, A., Reumers, V., Braeken, D., et al. (2018b). A multimodal cmos mea for high-throughput intracellular action potential measurements and impedance spectroscopy in drug-screening applications. *IEEE Journal of Solid-State Circuits*, (99):1–11.

Maccione, A., Gandolfo, M., Zordan, S., Amin, H., Di Marco, S., Nieus, T., Angotzi, G. N., and Berdondini, L. (2015). Microelectronics, bioinformatics and neurocomputation for massive neuronal recordings in brain circuits with large scale multielectrode array probes. *Brain research bulletin*, 119:118–126.

Maccione, A., Simi, A., Nieus, T., Gandolfo, M., Imfeld, K., Ferrea, E., Sernagor, E., and Berdondini, L. (2013). Sensing and actuating electrophysiological activity on brain tissue and neuronal cultures with a high-density cmos-mea. In *Solid-State Sensors, Actuators and Microsystems (TRANSDUCERS & EUROSENSORS XXVII), 2013 Transducers & Eurosensors XXVII: The 17th International Conference on*, pages 752–755. IEEE.

Maxwell Biosystems (2018). Maxone single-well mea. https://www.mxwbio.com/products/maxone-mea-system-microelectrode-array/. Accessed: 2018-10-31.

MCS (2018). Multi channel systems cmos-mea5000-system. https://www.multichannelsystems.com/products/cmos-mea5000-system. Accessed: 2018-10-31.

Müller, J., Bakkum, D. J., and Hierlemann, A. (2013). Sub-millisecond closed-loop feedback stimulation between arbitrary sets of individual neurons. *Frontiers in neural circuits*, 6:121.

Navajas, J., Barsakcioglu, D. Y., Eftekhar, A., Jackson, A., Constandinou, T. G., and Quiroga, R. Q. (2014). Minimum requirements for accurate and efficient real-time on-chip spike sorting. *Journal of neuroscience methods*, 230:51–64.

Nenadic, Z. and Burdick, J. W. (2005). Spike detection using the continuous wavelet transform. *IEEE Transactions on Biomedical Engineering*, 52(1):74–87.

Newman, J. P., Zeller-Townson, R., Fong, M.-F., Arcot Desai, S., Gross, R. E., and Potter, S. M. (2013). Closed-loop, multichannel experimentation using the open-source neurorighter electrophysiology platform. *Frontiers in neural circuits*, 6:98.

Nguyen, T. K. T., Navratilova, Z., Cabral, H., Wang, L., Gielen, G., Battaglia, F. P., and Bartic, C. (2014). Closed-loop optical neural stimulation based on a 32-channel low-noise recording system with online spike sorting. *Journal of neural engineering*, 11(4):046005.

Nicolelis, M. A., Fanselow, E. E., and Ghazanfar, A. A. (1997). Hebb's dream: the resurgence of cell assemblies. *Neuron*, 19(2):219–221.

Normann, R. A., Campbell, P. K., and Jones, K. E. (1993). Three-dimensional electrode device. US Patent 5,215,088.

Novellino, A., D'Angelo, P., Cozzi, L., Chiappalone, M., Sanguineti, V., and Martinoia, S. (2007). Connecting neurons to a mobile robot: an in vitro bidirectional neural interface. *Computational Intelligence and Neuroscience*, 2007.

Obien, M. E. J., Deligkaris, K., Bullmann, T., Bakkum, D. J., and Frey, U. (2015). Revealing neuronal function through microelectrode array recordings. *Frontiers in neuroscience*, 8:423.

Obien, M. E. J., Gong, W., Frey, U., and Bakkum, D. J. (2017). Cmos-based high-density microelectrode arrays: Technology and applications. In *Emerging Trends in Neuro Engineering and Neural Computation*, pages 3–39. Springer.

Oxford English Dictionary (2018). *OED online*. Oxford University Press. http://www.oed.com. Accessed: December 2018.

Pang, J.-J., Gao, F., and Wu, S. M. (2003). Light-evoked excitatory and inhibitory synaptic inputs to on and off $\alpha$ ganglion cells in the mouse retina. *Journal of Neuroscience*, 23(14):6063–6073.

Parent, A. (2004). Giovanni aldini: from animal electricity to human brain stimulation. *Canadian journal of neurological sciences*, 31(4):576–584.

Park, J., Kim, G., and Jung, S.-D. (2017). A 128-channel fpga-based real-time spike-sorting bidirectional closed-loop neural interface system. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25(12):2227–2238.

Piccolino, M. (1997). Luigi galvani and animal electricity: two centuries after the foundation of electrophysiology. *Trends in neurosciences*, 20(10):443–448.

Pine, J. (2006). A history of mea development. In *Advances in network electrophysiology*, pages 3–23. Springer.

Potter, S. M., El Hady, A., and Fetz, E. E. (2014). Closed-loop neuroscience and neuroengineering. *Frontiers in neural circuits*, 8:115.

Purves, D., Fitzpatrick, D., Hall, W., Augustine, G., and Lamantia, A. (2011). Neuroscience . sunderland, mass.: Sinauer. Technical report, ISBN 978-0-87893-695-3.

Quiroga, R. Q., Nadasdy, Z., and Ben-Shaul, Y. (2004). Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural computation*, 16(8):1661–1687.

Rey, H. G., Pedreira, C., and Quiroga, R. Q. (2015). Past, present and future of spike sorting techniques. *Brain research bulletin*, 119:106–117.

Rolston, J. D., Gross, R. E., and Potter, S. M. (2009). A low-cost multielectrode system for data acquisition enabling real-time closed-loop processing with rapid recovery from stimulation artifacts. *Frontiers in Neuroengineering*, 2:12.

Rolston, J. D., Gross, R. E., and Potter, S. M. (2010). Closed-loop, open-source electrophysiology. *Frontiers in neuroscience*, 4:31.

Scanziani, M. and Häusser, M. (2009). Electrophysiology in the age of light. *Nature*, 461(7266):930.

Seu, G. P., Angotzi, G. N., Boi, F., Raffo, L., Berdondini, L., and Meloni, P. (2018). Exploiting all programmable socs in neural signal analysis: A closed-loop control for large-scale cmos multielectrode arrays. *IEEE Transactions on Biomedical Circuits and Systems*.

Seu, G. P., Angotzi, G. N., Tuveri, G., Raffo, L., Berdondini, L., Maccione, A., and Meloni, P. (2017a). A closed-loop system for neural networks analysis through high density meas. In *Ph. D. Research in Microelectronics and Electronics (PRIME), 2017 13th Conference on*, pages 321–324. IEEE.

Seu, G. P., Angotzi, G. N., Tuveri, G., Raffo, L., Berdondini, L., Maccione, A., and Meloni, P. (2017b). On-fpga real-time processing of biological signals from high-density meas: a design space exploration. In *Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2017 IEEE International*, pages 175–183. IEEE.

Seymour, J. P., Wu, F., Wise, K. D., and Yoon, E. (2017). State-of-the-art mems and microsystem tools for brain research. *Microsystems & Nanoengineering*, 3:16066.

Shelley, M. W. (2009). *Frankenstein, or, The Modern Prometheus, 1818*. Engage Books, AD Classic.

Silicon Laboratories, I. (2013). Improving adc resolution by oversampling and averaging. https://www.silabs.com/documents/public/application-notes/an118.pdf. Online; accessed 24 October 2018.

Sohail, A. and Li, Z. (2018). *Computational Approaches in Biomedical Nano-Engineering*. John Wiley & Sons.

Stankovic, J. A. (1988). Misconceptions about real-time computing: A serious problem for next-generation systems. *Computer*, 21(10):10–19.

Stevenson, I. H. and Kording, K. P. (2011). How advances in neural recording affect data analysis. *Nature neuroscience*, 14(2):139.

Thomas Jr, C., Springer, P., Loeb, G., Berwald-Netter, Y., and Okun, L. (1972). A miniature microelectrode array to monitor the bioelectric activity of cultured cells. *Experimental cell research*, 74(1):61–66.

Trimberger, S. M. (2015). Three ages of fpgas: A retrospective on the first thirty years of fpga technology. *Proceedings of the IEEE*, 103(3):318–331.

Tsai, D., John, E., Chari, T., Yuste, R., and Shepard, K. (2015). High-channel-count, high-density microelectrode array for closed-loop investigation of neuronal networks. In *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*, pages 7510–7513. IEEE.

Tsai, D., Sawyer, D., Bradd, A., Yuste, R., and Shepard, K. L. (2017). A very large-scale microelectrode array for cellular-resolution electrophysiology. *Nature communications*, 8(1):1802.

Venkatraman, S., Elkabany, K., Long, J. D., Yao, Y., and Carmena, J. M. (2009). A system for neural recording and closed-loop intracortical microstimulation in awake rodents. *IEEE Transactions on Biomedical Engineering*, 56(1):15–22.

Verkhratsky, A., Krishtal, O. A., and Petersen, O. H. (2006). From galvani to patch clamp: the development of electrophysiology. *Pflügers Archiv*, 453(3):233–247.

Viswam, V., Bounik, R., Shadmani, A., Dragas, J., Obien, M., Müller, J., Chen, Y., and Hierlemann, A. (2017). High-density mapping of brain slices using a large multi-functional high-density cmos microelectrode array system. In *Solid-State Sensors, Actuators and Microsystems (TRANSDUCERS), 2017 19th International Conference on*, pages 135–138. IEEE.

Viswam, V., Dragas, J., Shadmani, A., Chen, Y., Stettler, A., Müller, J., and Hierlemann, A. (2016). 22.8 multi-functional microelectrode array system featuring 59,760 electrodes, 2048 electrophysiology channels, impedance and neurotransmitter measurement units. In *Solid-State Circuits Conference (ISSCC), 2016 IEEE International*, pages 394–396. IEEE.

Wallach, A., Eytan, D., Gal, A., Zrenner, C., and Marom, S. (2011). Neuronal response clamp. *Frontiers in neuroengineering*, 4:3.

Wise, K. D., Angell, J. B., and Starr, A. (1970). An integrated-circuit approach to extracellular microelectrodes. *IEEE Transactions on Biomedical Engineering*, BME-17(3):238–247.

Wright, J., Macefield, V. G., van Schaik, A., and Tapson, J. C. (2016). A review of control strategies in closed-loop neuroprosthetic systems. *Frontiers in neuroscience*, 10:312.

Xilinx (2018). Zynq$^{®}$-7000 soc family. https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html. Accessed: 2018-10-31.

Xinyu, L., Hong, W., Shan, L., Yan, C., and Li, S. (2017). Adaptive common average reference for in vivo multichannel local field potentials. *Biomedical Engineering Letters*, 7(1):7–15.

Zaher, S., Lonardoni, D., Boi, F., Seu, G. P., Angotzi, G. N., Meloni, P., and Berdondini, L. (2019). A closed-loop system processing high-density electrical recordings and visual stimuli to study retinal circuits properties. In *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)*. IEEE. [Accepted; To Be Published].

Zrenner, C., Eytan, D., Wallach, A., Thier, P., and Marom, S. (2010). A generic framework for real-time multi-channel neuronal signal analysis, telemetry control and sub-millisecond latency feedback generation. *Frontiers in neuroscience*, 4:173.