UNIVERSIDAD DE SEVILLA

DOCTORAL THESIS

Neuromorphic auditory computing: towards a digital, event-based implementation of the hearing sense for robotics

Author:

Daniel Gutiérrez Galán

Supervisors:

Dr. Alejandro Linares Barranco
Dr. Ángel F. Jiménez Fernández

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Robotics and Computer Technology Lab.
Departamento de Arquitectura y Tecnología de Computadores

July, 2022

Declaration of Authorship

I, Daniel Gutiérrez Galán, declare that this thesis, titled "Neuromorphic auditory computing: towards a digital, event-based implementation of the hearing sense for robotics", and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UNIVERSIDAD DE SEVILLA

Abstract

Escuela Politécnica Superior

Departamento de Arquitectura y Tecnología de Computadores

Doctor of Philosophy

Neuromorphic auditory computing: towards a digital, event-based implementation of the hearing sense for robotics

by Daniel Gutiérrez Galán

In this work, it is intended to advance on the development of the neuromorphic audio processing systems in robots through the implementation of an open-source neuromorphic cochlea, event-based models of primary auditory nuclei, and their potential use for real-time robotics applications.

First, the main gaps when working with neuromorphic cochleae were identified. Among them, the accessibility and usability of such sensors can be considered as a critical aspect. Silicon cochleae could not be as flexible as desired for some applications. However, FPGA-based sensors can be considered as an alternative for fast prototyping and proof-of-concept applications. Therefore, a software tool was implemented for generating open-source, user-configurable Neuromorphic Auditory Sensor models that can be deployed in any FPGA, removing the aforementioned barriers for the neuromorphic research community.

Next, the biological principles of the animals' auditory system were studied with the aim of continuing the development of the Neuromorphic Auditory Sensor. More specifically, the principles of binaural hearing were deeply studied for implementing event-based models to perform real-time sound source localization tasks. Two different approaches were followed to extract inter-aural time differences from event-based auditory signals. On the one hand, a digital, event-based design of the Jeffress model was implemented. On the other hand, a novel digital implementation of the Time Difference Encoder model was designed and implemented on FPGA.

Finally, three different robotic platforms were used for evaluating the performance of the proposed real-time neuromorphic audio processing architectures. An audio-guided central pattern generator was used to control a hexapod robot in real-time using spiking neural networks on SpiNNaker. Then, a sensory integration application was implemented combining sound source localization and obstacle avoidance for autonomous robots navigation. Lastly, the Neuromorphic Auditory Sensor was integrated within the iCub robotic platform, being the first time that an event-based cochlea is used in a humanoid robot. Then, the conclusions obtained are presented and new features and improvements are proposed for future works.

Acknowledgements

Becoming a Doctor of Philosophy has been one of my biggest dreams since I was young. The path has been long and the task has turned difficult, but there have been people all along this time who either have support me or have hated me. To those who have helped me in any kind of situation: thank you all very much. I have no words to let you all know how much I appreciate you. To those who have tried to bring me down: you were my biggest motivation. I have hundreds of words for those people, but only one that can be written here: "empathy".

I bought my first computer when I was 9 years old. All my friends asked for the Play Station 2. Instead, I asked for a Pentium 4 with 256 MBytes of RAM memory. Fransico José Junquero told me *"this is going to be the future, and you have the potential to learn whatever you want"*. Without that advice, I would not have asked for a computer, I would not have written these lines. Thank you very much, this thesis also belongs to you.

Then, I started to show interest for mathematics. When I was 14 years old, although my marks in math were good, my teacher D. Raúl Espinosa asked me *"What degree would you like to study at the University?"*, and I replied *"Computer Engineering"*. Without any doubt, he told me *"Man, your study methodology is shit. You must change it if you want to survive to the first year of university."* He was right, and my marks in math during those two years were horrible. Nevertheless, I learned to be self-critical and to identify when I was doing things wrong. Many thanks, Raúl.

Right after that period, in the year before moving to the university, I met one of the best teachers I have ever had: Carmelo. He taught me the best methodology to study any science-related subject. Nowadays, I still use his methodology also for this thesis. And because of that, studying math was not a nightmare anymore; instead, it was a game to be enjoyed. I will teach my students by using the same procedure. It will be an honour.

From the University of Seville, where I started to study Computer Engineering in 2014, I would like to thanks all the professors I had, because I took the best from all of them. In particular, I would like to thank David Ruíz Cortés, since he was the first who believed in me. And I also have to say sorry, since we started working together in big data but I realized I preferred hardware-related tasks. I will never forget our first meeting when you told me *"Fight to be the boss and then change every single thing that it is not fair here"*. I am on my way.

In addition, thanks to all my colleagues from the Department of Architecture and Technology of Computers, to which I belong since 2016, for all the advice, both scientist and non-scientist discussions, good and bad moments. You all became my family when I started to spend more than 10 hours per day in the lab. I will take the liberty of mentioning some colleagues who especially have shared more moments with me. Thank you, Ricardo and Antonio, for being good friends, Rafa, Elena, Fernando, Paco, Gabriel, Manuel Domínguez and Manuel Rivas, among others, for sharing with me your experience, and finally Juanma, and Alberto, for the technical support and your predisposition.

Although I tried to work in a small company, refusing my dream of working at the university, I felt that was not my place. It would be impossible to be the person who I am today without the confidence of Alejandro Linares Barranco, Ángel Jiménez Fernandez, who also are my supervisors, and Juan Pedro Domínguez Morales, who is my friend, my mentor, and the guy with who I would go to hell. Thank you very much Alejandro for giving me the opportunity to make my dream come true, letting me learn thousands of things at the best research centres and with the best researchers. In addition, thank you very much Ángel for being my inspiration and my reference, for supporting me and for being my friend. Both Alejandro and you have been the persons in charge to train me, and all my appreciation goes to you.

However, I have been able to write those lines because of Juan Pedro. Man, we have walked around in Manchester, Bielefeld, Berlin, Paris, Capo Caccia, Cádiz, and Seville. We have cried, we have worked during weekends and holidays... But the most important thing, you have remembered me to not give up. Thank you very much for everything. I would need half of this thesis just for mentioning every single thing you have helped me or shared with me.

During the years I have been PhD. student, I have travelled a lot. It is still incredible for me how much I have learned from all the places I have visited. I would say that I have learned more from every single visit rather than from the whole PhD. itself. The first event I attended as a PhD. student was to the Capo Caccia Neuromorphic Workshop in 2017. Two important events occurred there which were decisive for my career: 1) it was the first time I saw the iCub robot from the Istituto Italiano di Tecnologia; at that moment, I thought "I have to work with this robot"; 2) I met amazing people there who nowadays are great researchers and friends. I would like to say thanks to Charlotte Frenkel, Tim Walther, Alessandro Aimar, Enea Ceolini, Stefano Buccelli, Elisa Donati, James Knight, Moritz Milde, and Giacomo Pedretti for sharing with me those awesome two weeks. And also for saving my life during the "red wine saturation" episode.

There was only one problem at that moment: my English level was not enough to share my thoughts. The "English problem" was solved by Franca Oldenburg, our English teacher at the beginning, our best friend now. I could not be writing this document in English without your help. Thank you for being such a nice person, teacher, and friend.

Regarding to working with the iCub robot, the path was longer. I had the great opportunity to do a research visit to the SpiNNaker group at the University of Manchester, led by Professor Steve Furber, in 2017. I could work with Robert

James, Qiam Liu, Garibaldi Pineda Garcia, Gabriel Fonseca Guerra, Luís Plana, Andrew Rowley, Michael Hopkins, Alan Stokes and Simon Davidson. Thank you very much for letting me feel like a part of your team. Special mention to Jesko Meißel, one of my flatmates in Manchester, for opening my mind.

The most important turning point for my PhD. was in 2018, when I attended to the Barcelona Cognition, Brain and Technology Summer School. After the good experience in Capo Caccia, I attended to this summer school with high expectations. Once there, I realized it was not a neuromorphic summer school, instead the main topic was more focused on neuroscience. For those two weeks, several work groups were created, but I could not find any work group related to my research topic. Then, it happened: Adrián Fernández Amil told me about the sound localization procedure in humans, and I got fascinated. Thank you very very much for that conversation, it redirected my research topic. Furthermore, I met amazing people there: Joan, Diletta, Valeria, Giampiero, Jordi, Álvaro, and María. To all of you: thank you very much! You changed my file.

Right after the summer school, I went to Bielefeld to collaborate with the group of Prof. Elisabetta Chicca for two weeks. My good friend Tim Walther was working there, and he made it much easier. Thanks for the AquaFitness classes, for organizing dinners and for the awesome beer testing. Thanks also to Martin for all the good moments and coffee breaks. Especially, I would like to thank Thorben, with whom I have worked hard in different projects, for being such a nice colleague, and to Philipp, for being aware of me and tolerating the pure tone tests in his office. I will never forget the Jalapeños restaurant and its amazing Nachos.

In November 2018, Prof. Chiara Bartolozzi visited Seville for giving a talk. I met her at Capo Caccia Neuromorphic Workshop in 2017, when she presented a demo using iCub. Different from the Daniel that attended to that workshop, one year and a half later, I had enough knowledge to have a scientist discussion with Chiara. We talked about the possibility to include our neuromorphic audio sensor within the iCub, the potential applications of that, and how much I would like to work with her, her group, and the robot. Today, I can say that it was one of the best decision of my life. Thanks a lot, Chiara, for believing in me since the beginning, for helping me during the visits and for still being my mentor.

I could say that 2019 was my best year in terms of research. I went again to Bielefeld for two weeks, where I started a very nice project with Thorben and Juan Pedro that was afterwards presented at BioCAS conference. I got a fellowship from NEUROTECH for attending to the Capo Caccia Neuromorphic Workshop 2019. This time, both my English usage and my experience in the neuromorphic field were much better. I would like to thank Hector Gonzalez, Lyes Khacef, Arren Glover, Moritz Milde, Marco Monforte, Giulia D'Angelo, Omar Oubari, Melika Payvand, Ole Richter, Michael Schmuker, Baris Serhan, Beck Strohmer, Pau Vilimelis, Enea Ceolini, and Kathrin Aguilar for everything. We did not know, but that was the last time the contest was going to be held before COVID-19. And it was amazing!

Then, in July 2019 I moved to Genova for three months research visit at the Italian Instituto of Technology, with Prof. Chiara Bartolozzi. The visit would not have been possible without the help and perseverance of Marta Caracalli, who was fighting against the bureaucracy of both institutions. Thank you very much Marta for your help, your determination, and for your perfect Spanish. Also, thanks for being such a nice friend. And the project would not have been successfully finished without the help of Arren Glover, thanks to his advice and technical support, Maurizio Casti, available 24\7 for anything (thanks for the gifts after each visit!), Massimiliano Iacono and Marco Monforte, for making every single task easier, and Chiara Bartolozzi, for her support and useful advice.

But the biggest thing I got from that visit was the people I met there: my best friend Dennis, with who I lived one of the best summers ever, still remembering our last techno party in Milan; Ander, sharing hundreds of beers; again Marco and Massi, for all the hours we have spent together in the lab; Simon, Elisa, Viola, Shuman, Elena, Simeon, Ella, Giuseppe, Prashanth, Giulio, Damiano, Luca, Leandro, Juan, Raquel, Fabrizio, Christopher, Marlena, Guido, Tommaso, Virginia, and many more. You guys are my Italian family. Thank you very much.

On the personal side, I have a clear idea of the list of things I would like to thank. To the thousands of litters of coffee, for keeping me awake when I had too much work, and to the hundreds of litters of Jägermeister, for letting me forget all the bad moments. To Spotify, for 40000 minutes of music every year, providing me from techno music to classic music. To Steam, for stealing me hundreds of hours playing videogames instead of doing research. To Overleaf, for turning my account to the PRO version to write these lines after they felt my pain. To my friends, for remembering me every single day that I could be working in a big company earning a lot of money instead of doing shitty research.

And finally, to my family, my parents Isidoro and Regla, my brother Rubén, and my dog Arturo, for giving me their unconditional love and support, and for understanding all the sacrifices I did during these last five years. And especially to the persons I loved and love, for all the time separated by too many kilometres and the long conversations through video calls in which you all pushed me to keep working.

I know that this acknowledgements section has become longer, but both this thesis document and the current version of myself would not be possible and make no sense without every single person or situation mentioned in these lines. Thank you all for everything. You all made my life better.

Contents

List of Figures

List of Tables

List of Abbreviations

AAP	**A**uditory **A**scending **P**athway
ADC	**A**nalog-to-**D**igital **C**onverter
ADM	**A**synchronous **D**elta **M**odulation
AER	**A**dress **E**vent **R**epresentation
AGC	**A**utomatic **G**ain **C**ontrol
AI	**A**rtificial **I**ntelligence
ANN	**A**rtificial **N**eural **N**etwork
API	**A**pplication **P**rogramming **I**nterface
ASE	**A**daptative **S**ynaptic **E**fficacy
ASIC	**A**pplication-**S**pecific **I**ntegrated **C**ircuit
ATIS	**A**synchronous **T**ime-based **I**mage **S**ensor
AVCN	**A**ntero**V**entral **C**ochlear **N**ucleus
ASR	**A**utomatic **S**peech **R**ecognition
BW	**B**ack**w**ard
CAD	**C**omputer-**A**ided **D**esign
CAR	**C**ascade of **A**symmetric **R**esonators
CAR-FAC	**CAR** with **F**ast-**A**cting **C**ompression
CFB	**C**ascade **F**ilter **B**ank
CMOS	**C**omplementary **M**etal-**O**xide-**S**emiconductor
CN	**C**ochlear **N**ucleus
CPG	**C**entral **P**attern **G**enerator
CPU	**C**entral **P**rocessing **U**unit
CQFP	**C**eramic **Q**uad **F**lat **P**ackage
DAPGF	**D**ifferentiated **A**ll-**P**ole **G**ammatone **F**ilter
DCN	**D**orsal **C**ochlear **N**ucleus
DL	**D**elay **L**ine
DMWTA	**D**ecision **M**aking **W**inner-**T**ake-**A**ll
DOF	**D**egrees **O**f **F**reedom
DPI	**D**ifferential **P**air **I**ntegrator
DVS	**D**ynamic **V**ision **S**ensor
EDA	**E**lectronic **D**esign **A**utomation
eDVS	**e**mbedded **D**ynamic **V**ision **S**ensor
EPSC	**E**xcitatory **P**ost-**S**ynaptic **C**urrent
EPSP	**E**xcitatory **P**ost-**S**ynaptic **P**otential
ESM	**E**xhaustive **S**ynthetic **M**ethod
EU-SSG	**E**xhaustive **U**nsigned **S**ynthetic **S**pike **G**enerator

FAC	**F**ast-**A**cting **C**ompression
FFT	**F**ast-**F**ourier **T**ransform
FIFO	**F**irst **I**n. **F**irst **O**ut
FPAA	**F**ield-**P**rogrammable **A**nalog **A**rray
FPGA	**F**ield-**P**rogrammable **G**ate **A**rray
FSM	**F**inite **S**tate **M**achine
FW	**F**or**w**ard
GALS	**G**lobally **A**synchronous **L**ocally **S**ynchronous
GI	**G**lobal **I**nhibitory
GPU	**G**raphical **P**rocessing **U**unit
GUI	**G**raphical **U**ser **I**nterface
HDL	**H**ardware **D**escription **L**anguage
HPU	**H**ead **P**rocessing **U**nit
HRTF	**H**ead-**R**elated **T**ransfer **F**unction
HWR	**H**alf-**W**ave **R**ectifier
I2S	**I**ntegrated **I**nterchip **S**ound
IC	**I**nferior **C**olliculus
IHC	**I**nner **H**air **C**ells
IHL	**I**nside-**H**ead **L**ocalization
IIR	**I**nfinite **I**mpulse **R**esponse
IP	**I**ntellectual **P**roperty
IPSP	**I**nhibitory **P**ost-**S**ynaptic **P**otential
ILD	**I**nteraural **L**evel **D**ifference
ISI	**I**nter-**S**pike **I**nterval
ITD	**I**nteraural **T**ime **D**ifference
LFS	**L**atency to **F**irst **S**pike
LIF	**L**eaky **I**ntegrate-And-**F**ire
LSO	**L**ateral **S**uperior **O**live
LST	**L**ateral **S**ound **T**ransmitter
LUT	**L**ook-**U**p **T**able
MGB	**M**edial **G**eniculate **B**ody
MGN	**M**edial **G**eniculate **N**ucleus
MSO	**M**edial **S**uperior **O**live
MNTB	**M**edial **N**ucleus of the **T**rapezoid **B**ody
V_{mem}	**M**embrane **P**otential
M-SSG	**M**odulus **S**ynthetic **S**pikes **G**enerator
NAC	**N**euromorphic **A**uditory **C**omplex
NAS	**N**euromorphic **A**uditory **S**ensor
NAVIS	**N**euromorphic **A**uditory **VIS**ualizer
NoC	**N**etwork-**o**n-**C**hip
NSSOC	**N**euromorphic **S**pike-based **S**uperior **O**livary **C**omplex
OF	**O**ptical **F**low
OFE	**O**ptical **F**low **E**ncoder
OHC	**O**utter **H**air **C**ells

OZGF	**O**ne-**Z**erod **G**ammatone **F**ilter
PCB	**P**rinted **C**ircuit **B**oard
PDM	**P**ulse-**D**ensity **M**odulation
PFM	**P**ulse-**F**requency **M**odulation
PID	**P**roportional **I**ntegral and **D**erivative
PWM	**P**ulse-**W**idth **M**odulation
RAM	**R**andom **A**ccess **M**emory
RB-SSG	**R**everse **B**itwise **S**ynthetic **S**pikes **G**enerator
RIR	**R**oom **I**mpulse **R**esponse
ROM	**R**ead **O**nly **M**emory
RTL	**R**egister-**T**ransfer **L**evel
SAIF	**S**witching **A**ctivity **I**nterchange **F**ormat
SBPF	**S**pike-based **B**and-**P**ass **F**ilter
SC	**S**uperior **C**olliculus
SCFB	**S**pike-based **C**ascade **F**ilter **B**ank
SCPG	**S**piking **C**entral **P**attern **G**enerator
sEMD	**s**piking **E**lementary **M**otion **D**etector
SFB	**S**pike-based **F**ilter **B**ank
SH&F	**S**pike **H**old **&** **F**ire
SH&CF	**S**pike **H**old **&** **C**oincidence **F**ire
SI	**S**ensory **I**ntegration
SLPF	**S**pike **L**ow **P**ass **F**ilter
SNN	**S**piking **N**eural **N**etwork
SNR	**S**ignal-to-**N**oise **R**atio
SoC	**S**ystem **O**n a **C**hip
SOC	**S**uperior **O**livary **C**omplex
SPL	**S**ound **P**ressure **L**evel
SPTC	**SP**atio-**T**emporal **C**orrelation
SSD	**S**ound **S**ource **D**irection
SSP	**S**pike **S**ignal **P**rocessing
STDP	**S**pike-**T**iming-**D**ependent **P**lasticity
SC	**S**uperior **C**olliculus
TCL	**T**ool **C**ommand **L**anguage
TCP	**T**ransmission **C**ontrol **P**rotocol
TDE	**T**ime **D**ifference **E**ncoder
TSMC	**T**aiwan **S**emiconductor **M**anufacturing **C**ompany
VCN	**V**entral **C**ochlear **N**ucleus
VHDL	**V**HSIC **H**ardware **D**escription **L**anguage
VLSI	**V**ery **L**arge-**S**cale **I**ntegration
WPF	**W**indows **P**resentation **F**oundation
WTA	**W**inner-**T**ake-**A**ll
XML	e**X**tensible **M**arkup **L**anguage
YARP	**Y**et **A**nother **R**obot **P**latform

Dedicated to all the people who have been mentioned in the acknowledge.

Part I

Thesis

Chapter 1

Introduction

"To understand reality, you have to understand how things work. If you do that, you can start to do engineering with it, build things. And if you can't, whatever you're doing probably isn't good science."

– Carver Mead

Artificial visual processing is a current hot topic in the technological revolution that we are living in recent years. We have autonomous cars that are able to identify, understand and follow traffic signals, safely drive on a highway, and predict accidents even before the driver. Computer-based visual processing techniques are also used in factories to detect and discard faulty products in the assembly line. The advance is so impressive that even those techniques are being applied in medical fields for detecting diseases with more accuracy than specialized doctors.

The sense of vision provides humans with a large quantity of details that help us to understand our environment. This is why the vision could be considered the most important sense. Nevertheless, sometimes visual stimuli need to be put in context to fully understand reality. Among the whole set of human senses, the hearing sense is closely linked to the sight sense.

Speech allows us to communicate, interact and share ideas and feelings. It also helps us to make our daily task easier thanks to the voice assistants. The wide range of technologies employed in the development of such devices mixes speech recognition, natural language processing, and contextualized information search, among others. Commonly, the sound is sent to the cloud, where it is processed by high-performance servers applying digital signal processing techniques. Hence, we could say that the processing engine is centralized outside the device.

Although this approach has been proved to work, it faces some inconveniences which could be desirable to improve or remove, as power consumption during the "always-on" mode even if there is no data to process, the complexity of algorithms or noise robustness. Researchers have tried to solve those problems by applying many different techniques until they started to take inspiration from biology.

Species have been evolved over millions of years, slowly improving their body shapes, abilities and senses. In this process, the brain has played a key role by adapting itself according to the needs, optimizing task resolution while keeping a low power consumption. Since we could confirm that the brain is one of the most efficient computers ever made, it is the perfect candidate to be mimicked.

Humans are able to hear thanks to the ear, which receives sound waves, stimulating the inner ear, and sending the neuronal activity to the auditory cortex through a pathway. Differently to the cloud-based devices, sound's features are extracted in different brain nuclei across the auditory ascending pathway, being processed by the auditory cortex afterwards. The function carried out by each nucleus is generally specific, which does not mean that the process to do it is simple. And this thought could lead to follow phenomenological modelling of bio-inspired electronics devices instead of an emulation approach.

This way, many works can be found in the literature which aim to design, part by part, a human-like hearing sense, starting from the ear and its cochlea, continuing by the extraction of binaural cues for the spatial context in the early brain nuclei, and finishing with sound detection and speech recognition tasks in the auditory cortex. Despite those efforts, the use of a bio-inspired solution for solving auditory tasks is not as high as desired. Maybe because there is a gap between used technologies or approaches; maybe because the sound itself is not enough to take a decision and it needs to be exploited together with other sensory data.

But what is true is that the hearing sense is involved in all our daily tasks, from learning to teaching, from protecting to being safe. Therefore, to understand how it works and for applying its benefits to society, either directly by developing bio-inspired cochlear implants or indirectly by mean of home-assistant humanoids robots, this thesis has been carried out with dedication, excitement, and effort.

1.1 Motivation

Currently, auditory perception is becoming popular in different research areas. For instance, cochlear implants are considered as one of the best achievements in the bio-engineering field (Zeng et al., 2008; Wilson et al., 1991; Eshraghi et al., 2012). The performance of such devices, inspired in a certain way from biology, is incredibly high, making the life of impaired people better. The technology under the cochlear implants is based on an external digital signal processor connected to an array of electrodes, which is placed in the human cochlea. Nevertheless, the fact that the user needs to have an external device could be a problem in some daily aspects. With the new technologies, such as the neuromorphic engineering, new bio-inspired approaches for auditory sensors could be used for improving the current version of cochlear implants. And it could lead to the creation of a new generation of fully neuromorphic cochlear implants with better features.

Although all the efforts were put on cochlear models, there could be other cases in which the disease affect not directly the cochlea but also the following sections of the auditory ascending pathway. This means that the person could have problems for spatial sound source localization, which is an essential task in safety and socialization. Furthermore, auditory perception is closely linked to visual perception, being both of them mixed in the brain. As examples, visual information helps the auditory perception when a person is talking by reading its lips, and the auditory information helps the visual perception to focus the attention by localizing the sound in the space (Massaro and Stork, 1998; Molholm and Foxe, 2005; Khacef et al., 2020).

This neuromorphic approach of the hearing sense, besides to help humans, can be also applied to robotics. On the one hand, the low-latency nature of the neuromorphic systems could be exploited for autonomous navigation, using in combination both visual and auditory sensors. This way, an autonomous car would be able to detect moving objects that could be out of its field of view thanks to the sound, reacting to that event in the order of hundreds of microseconds. On the other hand, human-like robots are considered the best platform to test this new bio-inspired systems before moving to real experiments with humans. Their use helps to finely improve the hardware, the software and the final applications by comparing their behavior with the expected human behavior.

After having a deep look at the state-of-the-art, and after being identified several aspects related to the neuromorphic auditory perception which can be improved, this work have been motivated to contribute in the following three main blocks:

- Facilitate the access of a neuromorphic auditory sensor to the neuromorphic research community: even though there exist different options, like in-silico analog cochlea models and some others digital implementations, they are not so accessible due to whether their high price, low availability, or difficulty of being configured and used. For further improvements in the neuromorphic auditory perception field, an open source auditory sensor would be desirable that could offer the researchers a global platform for performing experiments.
- Continue with the development of our neuromorphic auditory sensor by adding new useful features: the cochlea model is just the first step in the auditory ascending pathway. Therefore, our final desire would be to have a complete digital, spike-based hearing sense starting from the cochlea up to the auditory cortex. In this path, the next step is the implementation of the brain nucleus where the binaural cues for the sound source localization

are extracted. In addition, current cochlea models does not implement this feature directly. Instead, software models have been developed as a complement with specific requirements, making it difficult to integrate the whole system in the same robotic platform.

- Integrate the auditory model within robotic platforms for providing them with the sense of hearing: related to the first item, it is not common to find neuromorphic cochlea models, either hardware or software, already integrated in robots, thus making it difficult to advance in the filed of real-time audio-guided robotics applications. The cochlea models are often used for generating event-based audio datasets or testing local neuromorphic audio applications. But it would be desirable to have a robotic platform that can provide both a neuromorphic sensor set (such as neuromorphic retinas, cochleas, motor control, etc) together with a unique software framework that allows the user to manage the input sensory events, avoiding to have multiple individual sensors managed by multiple individual software.

The proposed improvements and contributions presented in this thesis, that have been carried out with the best of my intentions, modesty, effort, and without the aim to have the absolute truth, can be summarized as:

- The implementation of an open source tool for automatically generating the design files of a technology-independent, digital neuromorphic cochlea model. Using the NAS as reference (developed by Ángel Jiménez Fernández in his thesis), this software tool is able to generate custom Hardware Description Language (HDL) files ready to be synthetized by FPGA's oriented tools without the need of being modified by the user. This way, it would not be necessary to have previous knowledge of HDL code. As advantage, the generated HDL code is platform independent, therefore it can be deployed in any FPGA from different vendors. In addition, the first open hardware NAS Application-Specific Integrated Circuit (ASIC) has been designed and fabricated to test the performance of our approach.
- The design, implementation, and test of a digital, spike-based model of the human Superior Olivary Complex (SOC) for performing sound source localization tasks. This model takes the output spiking information generated by the NAS as input, and is able to extract useful information about the localization of sounds in the space. Since it can be deployed into an FPGA, this model is also technology-independent, configurable and scalable. In addition, a new approach for the Interaural Time Difference (ITD) estimation has been proposed by using the concept of the Time Difference Encoder (TDE) neuron model as an alternative to the Jeffress model. This TDE model, as well as a proof-of-concept of its usage for the sound source lateralization task, have been designed and implemented in this work.

- The integration of the neuromorphic auditory sensor within robotic platforms and the demonstration of real-time neuromorphic audio processing applications. In particular, the NAS has been used in two application-specific robots and in one general purpose robot. First, the NAS was used as input sensor for modifying the movement pattern of an hexapod robot in real-time, where an internal Spiking Central Pattern Generator (SCPG) (implemented on SpiNNaker) was commanded by the input auditory stimuli. Second, the sound source localization information was combined with visual information in order to implement a real-time, audio-visual collision avoidance system. Third, both the NAS and the localization model were integrated within the iCub robot and its control framework for multi-modal sensory processing tasks and auditory perception algorithms development.

This thesis is part of the research career of the Robotics and Computer Technology group (RTC, TEP-108), to which the author belongs. This work is focused on and aligned with different tasks that are part of national research projects, which have served as funding, along with the Spanish Ministry of Education, Culture and Sports, thanks to a Formación del Personal Investigador scholarship. The knowledge, materials, code, and documents generated in this thesis have been made open-source (as far as possible) to be used by the research community due to it was funded with public money. These projects are:

- COFNET project: Sistema Cognitivo de Fusión Sensorial de Visión y Audio por Eventos (TEC2016-77785-P).
- NPP project: Neuromorphic Processor (P051-15/E03).
- NPP2 project: Neuromorphic Processor Project Phase 2 (P062-18/E03).
- MIND-ROB project: Percepción y Cognición Neuromórfica para Actuación Robótica de Alta Velocidad (PID2019-105556GB-C33).

1.2 Neuromorphic engineering

Moderns computers are based on the von Neumann architecture, which was proposed in 1945 by the computer scientist John von Neumann, and it is still being used nowadays. Over the years, most of the efforts from traditional computing companies have been focused on improving the speed of computation without taking too much care of the power consumption. However, the main problem resides on there is a mismatch between the speed of the memory access and the speed of the Central Processing Units (CPUs), leading to the so-called von Neumann bottleneck (Alex, 2009).

Furthermore, Moore's law is thought to come to an end in the next years due to physical constraints such as increasing thermal noise (Kish, 2002). These developments have triggered increasing research efforts for the design of alternative computational architectures which may complement and augment traditional von Neumann machines. One interesting architecture is the human brain, which can compute complex correlations in real-time with approximately 100 billion neurons and more than 100 trillion synapses while consuming only 20 watts (Rigden, 1996; Drubach, 2000; Moravec, 1998).

Researchers from the California Institute of Technology (Caltech) started to study how to build brain-inspired computers in the late 80s, led by Prof. Carver Mead. Their initial goal was to mimic the behavior of neurons in nervous systems by means of Very Large-Scale Integration (VLSI) analog circuits. According to Prof. Mead, *"to understand reality, you have to understand how things work. If you do that, you can start to do engineering with it, build things"*. Therefore, his contribution consisted in understanding biological neural systems through silicon implementations, which has inspired the field of analog neuromorphic circuits design (Mead, 1989).

Nowadays, the research interest in this topic has grown exponentially. Given the ability of the brain and the fundamentally different substrate (asynchronous operation, co-localization of memory and computation, full parallelism, etc.), neural computation is a promising source of inspiration. The capability to take complex decisions in real time by means of limited sensory data poses the basis for the development of a new generation of edge computing devices.

In the last 20 years, many neuromorphic platforms have appeared. Between them, we can find large-scale, analog neuromorphic processor, as DYNAPs (Moradi et al., 2017) or BrainScaleS (Schemmel et al., 2010), or digital approaches, as SpiNNaker (Furber et al., 2013a) or Loihi (Davies et al., 2018). Moreover, the development of neuromorphic sensors has gained in popularity across the years, trying to be positioned as a real alternative to conventional sensors. This way, we can find neuromorphic cameras that mimics the how the human eye works so-called neuromorphic retinas (Lichtsteiner et al., 2008; Serrano-Gotarredona and Linares-Barranco, 2013); neuromorphic auditory sensors that implements how the inner hear works (Chan et al., 2007; Jimenez-Fernandez et al., 2017; Yang et al., 2016; Xu et al., 2018b); neuromorphic motor control systems imitating how the muscles are controlled (Jimenez-Fernandez et al., 2012; Perez-Peña et al., 2013a; Gómez-Rodríguez et al., 2016; Linares-Barranco et al., 2020; Zhao et al., 2020); neuromorphic odor sensor emulating the sense of smelling (Chicca et al., 2014; Vanarse, 2020); and neuromorphic touch sensors that could be potentially used in prosthetic hands (Bartolozzi et al., 2007; Ward-Cherrier et al., 2020).

Those platforms have in common that they allow to deploy spiking neural network models which take a spiking neuron as a basic computing unit (Furber, 2016). Thus, for processing the spiking information provided by the neuromorphic sensors, researchers have aimed at modelling specific parts of the brain with Artificial Neural Networks (ANNs). For doing so, neural connectomics and neurophysiological data acquired from biology have been studied intensely to extract the most important characteristics of neural computation (such as low latency and low power consumption) by multidisciplinary research teams.

Therefore, we can define the modern neuromorphic engineering concept as a research field that is dedicated to design and develop artificial computing systems whose physical properties, structures, or representation of the information are based on the biological nervous system, and where researchers from many different research fields, as physics, mathematics, biology, engineering and even psychology work together.

1.2.1 Taking inspiration from the nervous system

Spanish scientists Santiago Ramón y Cajal received the Nobel Prize in Physiology or Medicine in 1906 for his work on the structure of the nervous system in recognition of his discoveries. He discovered that the brain is composed of a series of independent and interconnected cells, the neurons. His studies were made possible by advances in staining methods and microscopes. A reproduction of one of the Ramon y Cajal drawings is shown in Fig. 1.1.

In a neuron, three different parts can be distinguished: dendrites, soma, and axon. These three main parts are shown in Fig. 1.2 as a simplified scheme. Dendrites can be considered as the "inputs" of the neuron and their main function is to collect information from other neurons, which is then sent to the soma. The soma acts as the central processing unit and performs non-linear operations on the received information. The information received and processed modifies the membrane potential of the neuron and, if a specific threshold is reached, an electric pulse is generated, which will be sent through the axon (the "output" of the neuron) to other neurons that are connected to this one.

The connection between two neurons is known as the synapse (Johnston and Wu, 1994), and this connection has complex physiological characteristics. Hodgkin and Huxley (Hodgkin and Huxley, 1939) analyzed in 1939 the electrical behavior of an isolated neuron by studying how its sodium and potassium channels behaved. He demonstrated that the representation, communication, and processing of information in a neuron is made by means of small electric pulses in time, known as action potentials or spikes. The neuron that sends a spike through a synapse is called pre-synaptic neuron, whereas the one that receives the spike is called post-synaptic neuron.

In general, the resting potential of a neurons is about -70 mV. If the opening of the ion channel results in a net gain of positive charge across the membrane, the latter membrane is said to be depolarized, as the potential approaches zero. This process is called Excitatory Post-Synaptic Potential (EPSP), as it brings the neuron's potential closer to its firing threshold (about -55 mV). However, if the opening of the ion channel results in a net gain of negative charge, this moves

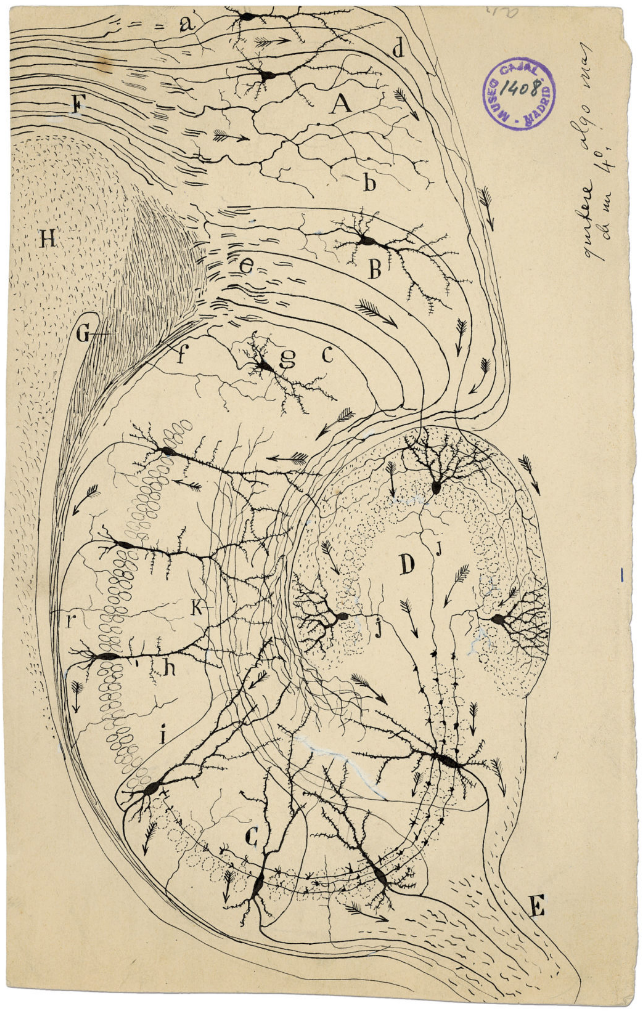


FIGURE 1.1: Cajal's drawing of neurons in the hippocampus, a region of the brain important to memory.

the potential further from zero and is termed hyperpolarization. This process is called Inhibitory Post-Synaptic Potential (IPSP), as it changes the charge across the membrane to be further from the firing threshold (Domínguez Morales, 2018). If the membrane potential of the neuron reaches the threshold, the neuron will depolarize abruptly, generating an action potential and transmitting a nervous impulse. After firing the spike and depolarizing the neuron, it will polarize up to the point of hyperpolarization, unable to emit a new spike until a specific time

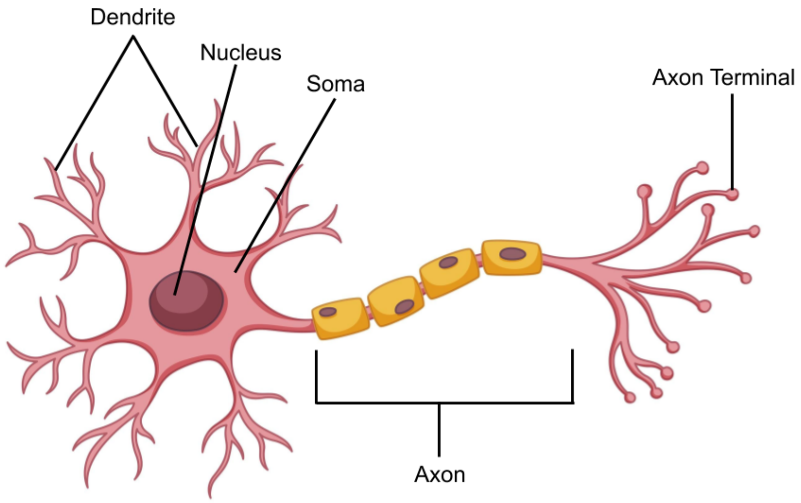


FIGURE 1.2: Simplified neuron anatomy.

period, known as the refractory period. Fig. 1.3. shows the diagram of an action potential generated by a neuron.

The hypothesis of how neurons represent the information (how the information is encoded into spikes) is one of the key aspects of neural processing research. Horace Barlow proposed different models (Barlow, 1961), including the one widely accepted within the neuromorphic engineering community. It proposes that the information is encoded in the frequency of the spikes using Pulse-Frequency Modulation (PFM) (Maass and Bishop, 2001; Westerman et al., 1997; Shepherd, 2003). In this way, the information can be encoded without the need to perform a temporal discretization of the information (Hynna and Boahen, 2001; Fujii et al., 1996). Other ways of encoding the information are through the ISI (Indiveri et al., 2006), or through the reset time, where the most important events are the ones that have been emitted first (Thorpe et al., 2010).

Some key features, such as simplicity, the reduction of the number of communication channels needed to transmit spikes, and the continuous flow of information in time, make the spiking representation of the information an efficient mechanism to work with neuron-based systems. Minimizing the number of channels needed for communication allows a high rate of interconnectivity between neurons. Therefore, since the information is not sampled, it avoids transmitting redundant information and saturating the communication channels in an unnecessary way by only transmitting spikes when they are needed.

This representation is also very robust to noise: the information is encoded

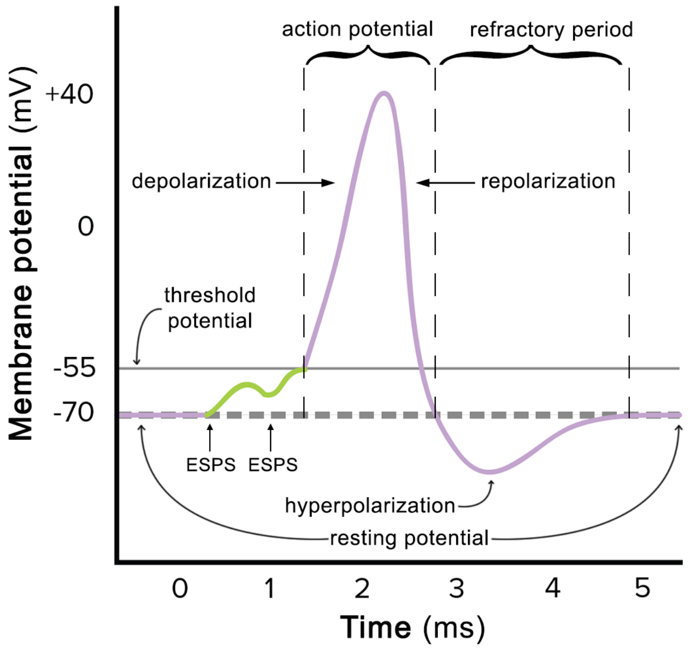


FIGURE 1.3: Diagram of a spike generated by a neuron (taken from (Domínguez Morales, 2018)).

in the time between two consecutive spikes (ISI), where it is only important if a spike exists or not. However, analog signals are completely immune to external perturbations.

The brain is the main part of the central nervous system in most of the animals. Often, it is located in the head, close to the most important sensory organs, and is protected by the cranium. The human brain is extremely complex and it is estimated to have around 100 billion neurons. In turn, one single neuron can be connected to another 10 thousand neurons. Neural structures are grouped into layers (mainly in the cortex because many nuclei are not layered) with the aim of processing part of the information obtained from input. Each layer has a specific functionality (Shadlen and Newsome, 1994; Rakic, 1988).

Performing a qualitative comparison between how a neural system and a digital computer work, many essential differences can be found:

- Asynchronous vs. global clock-commanded execution: computers are controlled by a global clock signal which frequency is generally set to a couple of gigahertz, and which makes the computer to update its state continuously every clock cycle. However, neurons process the information asynchronously without the need of any synchronization mechanism, thus avoiding the global clock signal. Some works suggest that the brain could have some synchronizing processes (Fries, 2005), although it has not been proven yet.

- Nondeterministic vs. deterministic: computers are completely deterministic elements, whose future internal state can be determined according to the set of instructions to be executed, while neurons respond to a stochastic model, depending on their reaction to their dynamic probabilistic models.
- No sampling rate vs. sampled data: computers work with high-resolution data that are sampled at a constant rate, whereas neurons are the exact opposite. In addition, they are able to adapt the characteristics of the information in real time to improve its representation.
- Decentralized vs. centralized processing: In most computational systems, processing is centralized and carried out by the CPU, in contrast to the way in which neurons process the information, where each neuron is considered a tiny CPU that processes a small part of the information independently of other neurons, thus being equivalent to a massively parallel computer.
- Large vs. low memory needs: computers need memory units to store instructions that will be executed and also for the data to be used by the instructions. However, neural systems do not need memory for any of these purposes, since the neural algorithms that they execute are encoded in the connection between different neurons. In other words, the information processing is in the spike stream itself.

1.2.2 From biology to engineering

Neural systems possess remarkable features that have motivated researchers to develop fully neuro-inspired devices. Taking as reference the biophysical behavioral description of a neuron, that models the neuron's behavior by using complex, non-linear mathematical equations, electronics engineers started to design analog circuits with the aim of to obtain a response similar to a neuron response. In its basic form, a neuron can be modelled as a resistor-capacitor circuit connected in parallel. Nevertheless, there exist tens of neuron's models in the living beings which have particular characteristics, thus needing more precise and specialized circuits designs to exactly reproduce their behaviors. Fig. 1.4 shows one example of circuit design modelling a retina photoreceptor, that can be considered as the first neuronal analog circuit model.

Analog circuits have significant similarities with neurons such as continuous electrical signals, asynchronous processing, and adaptive behavior, among others. Thus, analog neuron designs are considered a good approach when implementing neuromorphic artificial neuron models. Many works can be found in the literature presenting analog implementations of neuron models in the last

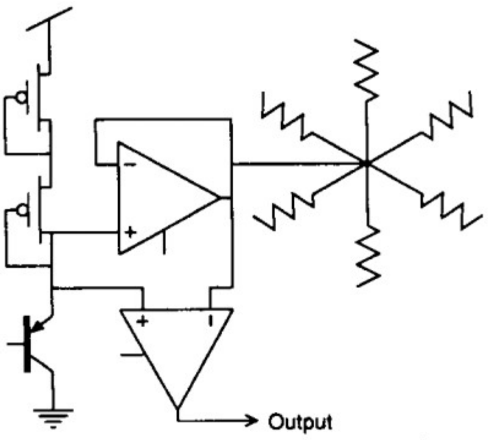


FIGURE 1.4: Schematic of pixel circuit from the Mahowald retina model (taken from (Mead, 1990)

30 years either using biophysical or phenomenological approaches (Bartolozzi and Indiveri, 2007; Brette and Gerstner, 2005; Milde et al., 2018). One of the most popular and implemented model is the LIF neuron (Liu et al., 2002; Indiveri et al., 2011a), that basically integrates the input current into the neuron's membrane potential until it reaches a certain threshold, producing then an output spike. In addition, this model takes into account the leak of the membrane potential voltage over time, reflecting the diffusion of ions through the membrane.

This kind of circuits generally consist of tens of transistors and a couple of capacitors, and those transistors can work at either sub-threshold or above-threshold, thus becoming very low-power consumption devices. These features have allowed to design analog neuromorphic procesors, as DYNAP(Moradi et al., 2017), and analog neuromorphic processing machines, as BrainScaleS (Schemmel et al., 2010). Moreover, researchers have turned bugs into features, as the device mismatch of analog circuits, by using the mismatch's randomness to increase the robustness of the neuron's learning procedure.

Nevertheless, digital implementations have been also carried out by following different approaches. On the one hand, the neuron's model can be implemented as a digital circuit to be later deployed into an FPGA or integrated into an ASIC. These circuits, in turn, can implement single or multiple neurons by following either biophysical (Cassidy and Andreou, 2008; Perez-Peña et al., 2019; Tapiador-Morales et al., 2018) or phenomenological (Frenkel et al., 2017; Indiveri et al., 2011b) modelling techniques. In addition, these neuron models can be used as processing unit of a neuromorphic processor, creating thus a Network-on-Chip (NoC) and allowing to implement large neural networks. MorphIC (Frenkel et al., 2019), Loihi (Davies et al., 2018), and SpiNNaker (Furber et al., 2013b) are good examples of massively multicore neuromorphic processors, among others.

On the other hand, neuron's models can be simulated by software algorithms which can be run in parallel computers or Graphical Processing Units (GPUs), taking advantage from their inherent parallel architecture (Knight and Nowotny, 2019; Knight and Nowotny, 2021). This way, large-scale population of neurons can be simulated, somethings even obtaining better performance when compared with neuromorphic hardware solutions, like SpiNNaker (Knight and Nowotny, 2018).

SpiNNaker (Spiking Neural Network Architecture) is a massively parallel multicore computing system to model very large SNNs in real time, optimized for neuromorphic applications. Both the system architecture and the design of the SpiNNaker chip have been developed by the Advanced Processor Technologies Research Group (APT) at the University of Manchester. Each SpiNNaker chip consists of eighteen 200 MHz general-purpose ARM968 cores, each one with its own memory. The chip contains a Globally Asynchronous Locally Synchronous (GALS) architecture with an asynchronous packet switching network that is highly optimized for neuromorphic applications (Plana et al., 2007). The communication between them is done via packets carried by a custom interconnect fabric. The transmission of these packets is managed entirely by hardware, giving the overall engine and extremely high bisection bandwidth. It is important to mention that one of the 18 ARM processors is used for management, and another ARM core is reserved. Therefore, only 16 ARM cores are involved in the neuromorphic process.

Fig. 1.5 bottom right shows a picture of the SpiNN-5 machine, which has 48 SpiNNaker chips (86 ARM processor cores) (shown in Fig. 1.5 bottom left) and 3 Spartan-6 FPGAs to communicate to other boards. Both boards have a 100 Mbps Ethernet connection that is used as control and I/O interface between the computer and the SpiNNaker board. SpiNN-5 has been used to build a million core massively parallel computer for human brain simulation (Furber and Brown, 2009), shown in Fig. 1.5 top.

In general, neuromorphic systems are composed of a neuromorphic sensor (or a group of them) and a set of spiking neural networks with multiple layers. These networks process the information provided by the sensors in the same way that the brain does. However, due to the brain has a high density in terms of neuronal connections, where each of the $10^5/\text{mm}^3$ neurons could be connected to other 10 thousand generating a density of connections up to $4\text{km}/\text{mm}^3$ (Braitenberg and Schüz, 1998), it is a hard task to design and fabricate this structure in a VLSI system. Nevertheless, a neuron presents a firing rate that ranges between 1-10 Hz, scaling up to kHz or MHz if many hundreds are combined together, meaning that state-of-the-art electronic circuits are much faster than biological neurons.

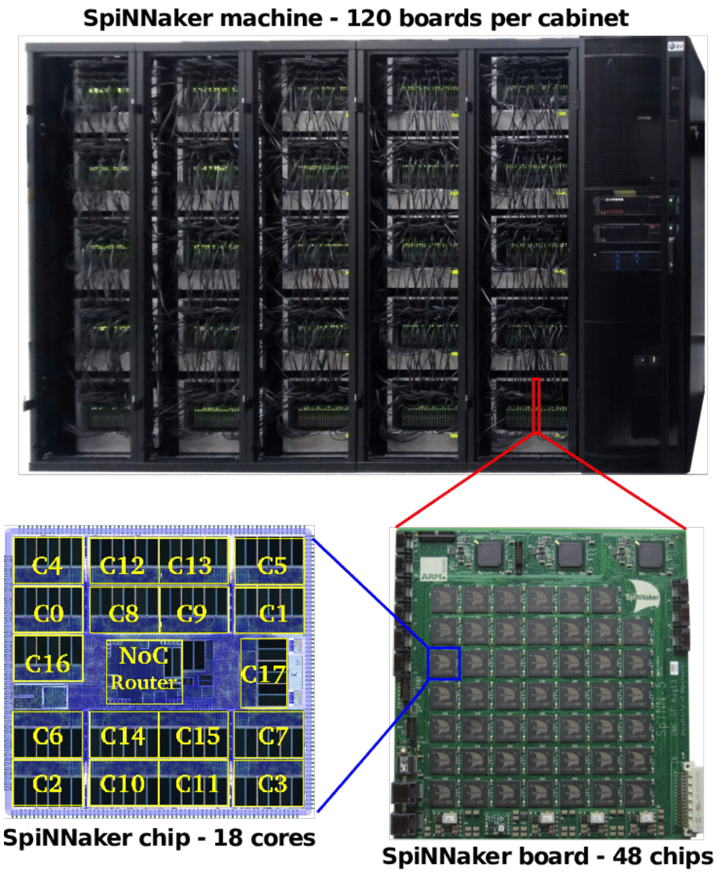

SpiNNaker machine - 120 boards per cabinet
C4
C12
C13
C5
C0
C8
C9
C1
C16
NoC Router
C17
C6
C14
C15
C7
C2
C10
C11
C3
SpiNNaker chip - 18 cores
SpiNNaker board - 48 chips

FIGURE 1.5: SpiNNaker machine project from chip to cabinete (taken from (Sugiarto et al., 2017)

A mechanism that multiplexes the information in time for a set of neurons in a single communication channel was proposed for the first time in 1991 (Sivilotti, 1991; Lazzaro et al., 1993; Lazzaro and Wawrzynek, 1995; Boahen, 2000) with the aim of solving connectivity problems, and it is known as AER. Based on this time difference and the high bandwidth capacity of the VLSI systems, each neuron is identified with a unique address, which is sent through a shared bus whenever the neuron fires a spike. It is an event-driven communication protocol used originally for transferring spikes (action potentials) between neurons in VLSI implementations (Mahowald, 1992). Thus, it is an event-driven asynchronous and digital multiplexing technique.

The main functionality of AER circuits is to provide multiplexing / demultiplexing mechanisms for spikes that are generated by / on sent to a set of neurons. Fig. 1.6 schematically shows the transmission of information between two neuromorphic chips using the AER protocol. This protocol uses a shared multiplexed high-speed bus (AER bus) for transmitting the spikes that are fired by the neurons on a chip. Each neuron is identified with a unique address. Every time that a neuron spikes, and thanks to an arbiter circuit, the address of that neuron will be placed in the AER bus, generating an AER event. This way, each of the asynchronous spikes will be encoded and multiplexed by the AER circuit in the bus in the same order as they were generated (Serrano-Gotarredona et al., 2009; Berge and Hafliger, 2007). The timestamp in which the address of the neuron is generated corresponds to the timestamp in which it was fired plus a small delay caused by the codification process. These coding circuits use a specific arbitrating logic to handle the transmission of multiple spikes simultaneously from different neurons. Many different AER encoders can be found in the literature, based on the mechanism used to solve the conflict of multiple simultaneous spikes and how the addresses are encoded (Cerezuela-Escudero et al., 2013). Each of these options have their own advantages and disadvantages depending on the neuromorphic system used. A comprehensive study of these mechanism can be found in (Liu et al., 2015).

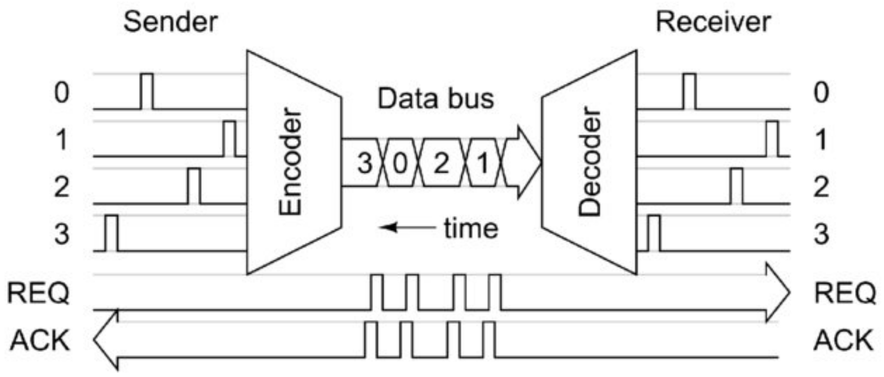


FIGURE 1.6: Transmission of an event using AER representation. Image taken from (Lazzaro et al., 1993).

The address of the neuron that produces the AER event is decoded on the receiver chip as soon as it arrives using an asynchronous decoder, sending the initial spike to the corresponding neuron. In this way, transmitter and receiver neurons are virtually connected through the AER bus using the AER protocol. If the delay between adjacent spikes in the input is high enough, the AER decoder will send the spikes to the corresponding neurons with a delay that corresponds to the time that the spike takes to reach the decoder plus a small delay caused by the decoding circuit.

The AER protocol is a 4-phase asynchronous handshake protocol between the transmitter and the receiver that guarantees the synchronization between both chips, as shown in Fig. 1.6 bottom. The transmitter starts the communication process with a request. Then, the second chip, the receiver, answers to this request with an acknowledgement. To conclude the transmission, the transmitter removes the request and the receiver does the same with the acknowledgement, resetting the system to its initial condition. Both parts of the communication are not active until the transmitter starts a new request. A new transmission depends on the neurons of the transmitter chip that try to send and AER event. Thus, AER is a data-driven protocol. In this asynchronous protocol, the activity within the communication bus depends on the data transmissions, which is in contrast with classical discrete and periodic systems.

With the increasing number of neuromorphic hardware including sensors and systems for deploying and running SNNs in real time, a proper set of software tools to process spiking information have become very useful for debugging tasks, dataset collection, and post-processing algorithms execution.

One of the most popular software tool by the neuromorphic community is jAER (Delbruck, 2008). It is an open-source (under GNU Lesser General Public License v2.1) framework for PCs for visualization of real-time or recorded event-based data[1], and rapid development of real-time event-based algorithms and applications built in Java. jAER consists of an application called "jAERViewer" that allows to plug in any AER device with USB interface and perform different functionalities with it, e.g. view the events coming from the device in real time, log (record) them to a file (with .aedat extension), play a logged AER stream back and process the events using different filters. Fig. 1.7 shows a screenshot from AER, receiving spikes in real time from the live output of a Neuromorphic Auditory Sensor (NAS).

There are many AER devices compatible with jAER, from dynamic vision and audio sensors to AER monitor/sequencer boards, along with a servo motor controller, among others. The events are produced by sensors asynchronously and timestamped (with 1 μs precision). Then, they are transmitted through the USB to the PC in packets, which contain variable numbers of events, and when they are received, jAER applies a set of filters chosen by the user. Meanwhile, the software can render the events that are output by the final filter and add visual annotations over the output.

For offline representation and analysis of .aedat files, Neuromorphic Auditory VISualizer (NAVIS) tool was proposed in (Dominguez-Morales et al., 2017c) as a complement of jAER. NAVIS is open-source tool [2] (under GNU General Public License) for post-processing the output spiking information from neuromorphic auditory sensors previously recorded with either Matlab or jAER. It allows you to load an events file and generate its full raster plot, so-

[1]With "event-based data" we mean address-events from systems using AER protocol which have been timestamped.

[2]NAVIS's project and code can be found in `https://github.com/jpdominguez/NAVIS-Tool`

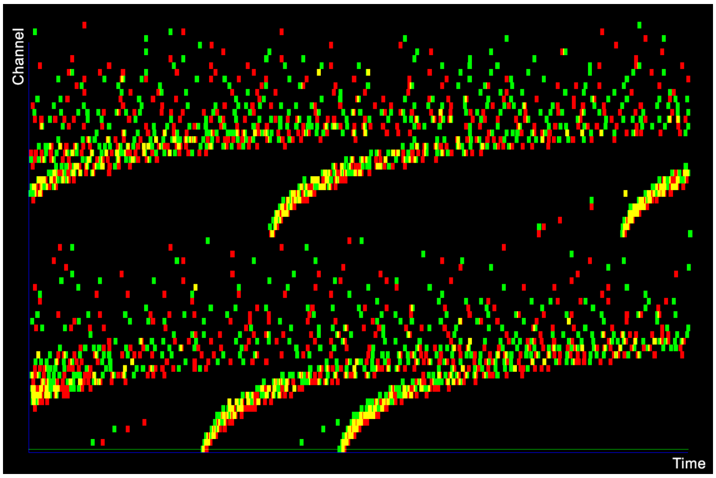


FIGURE 1.7: Screenshot of NAVIS tool.

called spikegram, histogram, sonogram, disparity, and mean activity plots. A spikegram example from NAVIS is shown in Fig. 1.8.

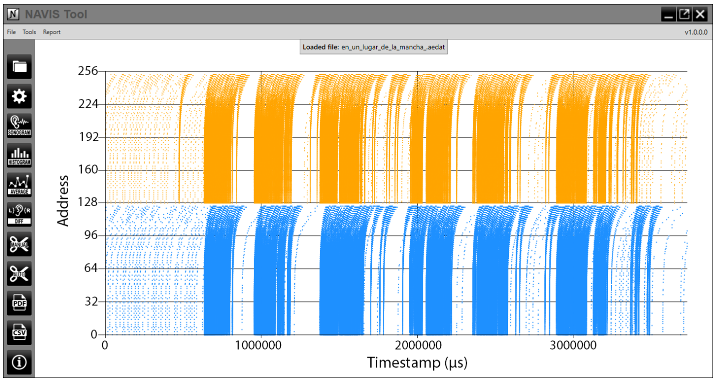


FIGURE 1.8: Main window of NAVIS tool showing the cochleogram of the sentence "En un lugar de La Mancha". Blue dots represent the output of the left 64-channels, while orange dots represent the output for the right channels. Image taken from (Dominguez-Morales et al., 2017c).

Furthermore, this tool includes some options for generating audio datasets, like an automatic splitter or mono-to-stereo/stereo-to-mono converters, among others. NAVIS has been succesfully used, for instance, in (Dominguez-Morales et al., 2018a), where authors first converted an audio dataset from heart recordings to spikes using NAS and jAER, and then NAVIS for generating the sonogram of each recorded .aedat file.

Since NAVIS was implemented by using Windows Presentation Foundation (WPF), non-Windows users were not able to use it. Therefore, a cross-platform, open-source (under GNU General Public License) python version was developed by Dominguez-Morales et al. called pyNAVIS (Dominguez-Morales et al., 2021b) [3]. As opposed to NAVIS, pyNAVIS does not have Graphical User Interface (GUI), so it is not as much interactive as NAS could be due to every action to be carried out must be already included in the script. However, the advantage of pyNAVIS is that it can be integrated into bigger scripts as a package, thus allowing the user to do multiple task but using the same Python script.

1.3 The sense of hearing

In previous chapters, neuro-inspired systems were introduced. These systems mimic the way in which the senses and the brain process the information. Thus, it is intended to obtain benefits in the processing that are present in living organisms and that could be used for particular tasks like speech recognition and sound source localization. To be able to develop this kind of systems, it is necessary to have basic biological knowledge of the processing to be emulated. Therefore, in this chapter, the main characteristics of the sense of hearing are presented, as well as a state-of-the-art analysis of different bio-inspired neuromorphic auditory systems that have already been published.

1.3.1 Auditory system in biology

In this chapter, the anatomy and the physiology of the auditory system are described, emphasizing the parts and structures that are more relevant for audio processing and sound source localization.

The concept of perception consists in the detection of a stimulus by one or more sensory receptors. In addition, interpretation, elaboration, and attentional selection are also components of perception. The sensory receptors of the human body are continuously sensing things that we are not even aware of (e.g., blood pressure, blood temperature, carbon dioxide and oxygen concentration, etc.) due to the fact that the signals that are sensed are not directly sent to the ensemble of network interactions that give rise to consciousness. In other circumstances, perception leads to sensation (or conscious perception). In these cases, the sensory receptors transmit impulses to the brain cortex through nerves and pathways. The brain cortex is able to integrate these signals into a sensation in the order of tens or hundreds milliseconds (Hull, 2011).

Hearing is one of the senses that are classified as special senses, which have specialized organs devoted to them (vision has the eyes, hearing has the ears, smell has the nose, and taste has the tongue). Therefore, hearing could be

[3]pyNAVIS's project and code can be found in `https://github.com/jpdominguez/pyNAVIS`

defined as the detection of sound waves and their integration in order to generate sensations (Hull, 2011).

Every sensation, including the ones produced by hearing, are the result of the same sequence of events: first, a stimulus is produced. Then, the sensory receptor detects the stimulus and converts it into an electrical signal. Next, the signal is transmitted to the brain (through the auditory pathways in the case of hearing). Finally, the brain integrates the signal into conscious perception (sensation).

The task of receiving the audio signal, processing it and transforming it into impulses are carried out inside the ear, while neural processing and sound recognition are performed in the brain. Thus, two main regions or sections can be distinguished in the auditory system: the peripheral auditory system, where sound waves are converted into nerve impulses, and the central auditory nervous system, which transforms these impulses into sensations.

Different cognitive processes intervene in the central auditory nervous system, by which context and meaning are given to the sound, i.e., they allow the recognition of words, the classification of different musical instruments, the localization of the sound source, distinguishing between different speakers by their pitch and loudness, etc., among other complex tasks.

1.3.1.1 The ear

The peripheral auditory system, also known as the ear, is responsible for the physiological processes of hearing. These processes allow the reception of sound, transforming it into electrical impulses that are then sent to the brain through the auditory nerves. The mechanical processing of the sound waves and their transformation into impulses are non-linear processes (Zwicker and Fastl, 2013), which hinders the characterization and modelling of the auditory perception.

The peripheral auditory system is divided into three interconnected parts: the outer ear, the middle ear, and the inner ear. Next, the anatomy and functionalities of each of these three parts of the ear are described, along with the propagation and the processing of the sound across them.

1.3.1.1.1 Outer ear

The outer ear is the external part of the ear and it consists of the auricle and the ear canal. The auricle is the visible part of the ear that resides outside the head (the word "ear" is also used to refer to this part alone) and it is also known as pinna. Its function is to gather the sound waves and guide them through the ear canal. When they hit the auricle, the sound waves are reflected and attenuated, which provides additional information to the brain for determining the direction of the sound that the auricle is receiving. The determination of the direction of the sound depends on the outer ear anatomy. Except for specific cases in

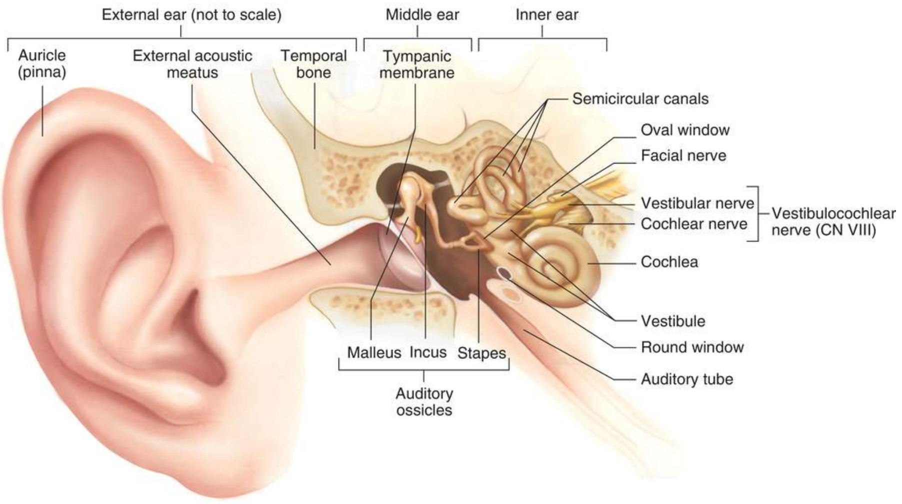


FIGURE 1.9: Anatomy of the ear. Image taken from (Patton et al., 2012).

which sounds come directly from towards, rearwards, above or below, the sound reaches the closest ear in a fraction of a second earlier and louder than in the furthest one. If the difference is higher than 10 μs (minimum human interaural time difference threshold), the brain is able to detect and interpret it.

The ear canal is 2.5 cm long and it extends from the auricle to the eardrum, also called the tympanic membrane, in the middle ear. Its main function is to protect the middle ear and to maintain the middle ear at a stable temperature. It conducts the vibrations gathered in the auricle to the tympanic cavity, amplifying sounds with frequencies between 3 and 12 KHz. Each of these components are shown in Fig. 1.9.

1.3.1.1.2 Middle ear

The middle ear is the part of the ear that is internal to the tympanic membrane, which separates this part from the outer ear (Fig. 1.9). The eardrum vibrates when it is hit by sound waves. The middle ear contains three small ossicles (malleus, incus, and stapes), which propagate the sound waves from the eardrum to the inner ear. The stapes are in contact with one of the fluids contained in the inner ear through the oval window. Thus, the ossicular chain acts as a mechanism to transform air vibrations into waves in the fluid and membranes of the inner ear. In order to achieve this, the air pressure inside the middle ear must be the same as the atmospheric pressure, which is possible thanks to the Eustachian tube (also known as auditory tube or pharyngotympanic tube), connecting the middle ear with the nasopharynx and allowing the pressure to equalize between the middle ear and throat (Hull, 2011; Patton et al., 2012).

The main functions of the middle ear are:

- To increase the pressure received by the eardrum (impedance matching). This is essential, since the cochlea is full of liquid, instead of air, and the density and compressibility of the cochlear liquid (the perilymph) is almost four thousand times lower than that of the air. If we did not have a mechanism to increase the pressure inside the middle ear, only 0.1% of the tympanic pressure would reach the cochlea. This acts as a sound normalization step phase. To protect the inner ear structures from extremely loud sounds, the stapedius (the muscle that stabilizes the stapes), stiffens the ossicular chain by pulling the stapes away from the oval window of the cochlea, decreasing the transmission of vibrational energy to the cochlea for sounds below 1-2 kHz and above 85-90 dB. This mechanism is known as the stapedius reflex or acoustic reflex.
- To reduce the transmission of low-frequency sounds, acting as a low-pass filter, with an attenuation of 15 dB per octave in the 1 kHz band.

1.3.1.1.3 Inner ear

The inner ear is the innermost part of the peripheral auditory system. It consists of the vestibular system, which is dedicated to control balance, and the cochlea, dedicated to hearing.

The cochlea, which is shown in Fig. 1.10 top left, is 32-35 mm long, 4 mm^2 to 1 mm^2 wide (from the base to the apex) spiral-shaped cavity filled with two different fluids. Fig. 1.10 shows a cross section of the cochlea in which three tubular ducts can be seen. The central one is the cochlear duct (also known as scala media) and contains endolymph. The second and third ducts, called vestibular duct and tympanic duct (or scala vestibuli and scala tympani, respectively), contain the same fluid, perilymph, due to the fact that they are interconnected through a small opening in the apex of the cochlea called helicotrema. The base of the stapes is in contact with the fluid of the vestibular duct through the oval window, while the tympanic duct terminates at the round window in the tympanic cavity, as shown in Fig. 1.9. The lining between the cochlear duct and the vestibular duct is known as the Reissner's membrane, while the lining between the cochlear duct and the tympanic duct is called basilar membrane (Fig. 1.10) (Hull, 2011).

The basilar membrane is a structure whose width and rigidity are not constant: it is broad and rigid near the oval window, and it gets thinner and more flexible near the apex of the cochlea. Rigidity decays almost exponentially with the distance from the oval window. This variation affects the propagation

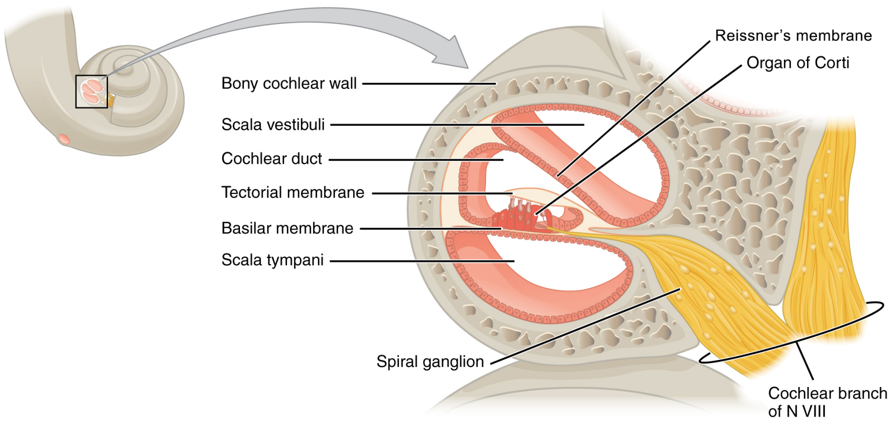


FIGURE 1.10: Detailed cross section of the cochlea.

speed of the sound waves across the basilar membrane, and it is responsible for one of the most important functionalities of the inner ear: frequency selectivity.

The basilar membrane supports the Organ of Corti (also known as the spiral organ), the most important element in the cochlea, which is able to transduct from movement into nerve impulses' action potential (see Fig. 1.11). The Organ of Corti contains between 15000 and 30000 receptors called hair cells. Their name derives from the tufts of stereocilia, known as hair bundles, which protrude from the apical surface of the cell into the cochlear duct. Each of the "hairs" of these cells are able to produce receptor potentials when they touch the tectorial membrane, which is located above the basilar membrane.

Hair cells can be divided into two different types: the Inner Hair Cells (IHCs) and the Outer Hair Cells (OHCs). There exist around 3500 IHCs and 20000 OHCs. Both cell types have connections with the afferent nerve fibers, which propagate action potentials towards the brain, and efferent nerve fibers, which propagate action potentials from the brain to the cochlea. However, the fiber distribution is very unequal: more than 90% of the afferent fibers innervate IHCs, whereas most of the 500 efferent fibers innervate OHCs. The functionality of each type is presented next.

The behavior of the cochlea starts with the vibrations produced by the stapes, which generates vibrations in the fluid contained in the vestibular duct. Oscillations in the perilymph of the vestibular duct are transmitted to the endolymph, and then to the basilar membrane, which also propagates the oscillations to the fluid in the tympanic duct (see Fig. 1.12). It is important to note that the amplitude and frequency of the vibrations are directly proportional to the amplitude and frequency of the sound waves.

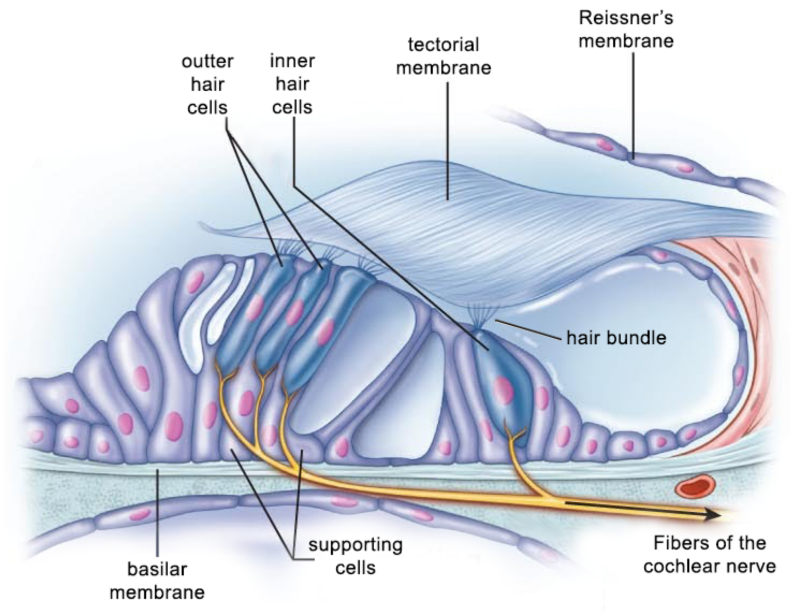


FIGURE 1.11: Inner structure of the Organ of Corti.

The sound wave that generates these oscillations has a peak value in its amplitude in a specific region of the cochlea that depends on the frequency of the wave and it tends to decrease rapidly near the apex. The lower the frequency of the sound is, the greater the distance that the wave will travel across the membrane before being attenuated, and vice versa. This way, the basilar membrane scatters the different frequency components of a complex spectrum signal in well-defined positions with respect to the oval window, as is shown in Fig. 1.13, known as a tonotopic maps.

The waves that are propagated through the cochlea produce a force or pressure to the cochlear duct, and therefore to the Organ of Corti. In the Organ of Corti, the hair cells rest upon the basilar membrane, and the external parts of the stereocilia are in contact with the tectorial membrane (see Fig. 1.11). Both membranes have different flexibility, making each of them move in relation to the other one when a wave travels across them. As a result, the external part of the hair cells bend when the membranes move. This bending produces a change in the action potential of the hair cells: depending on the bending direction, the hair cell hyperpolarizes or depolarizes. These variations in the action potential produce changes in the neurotransmitter released by the hair cells in the synapse with a first order neuron. The molecules of the neurotransmitters released in the synapse change the action potential of a neuron of the vestibulocochlear nerve,

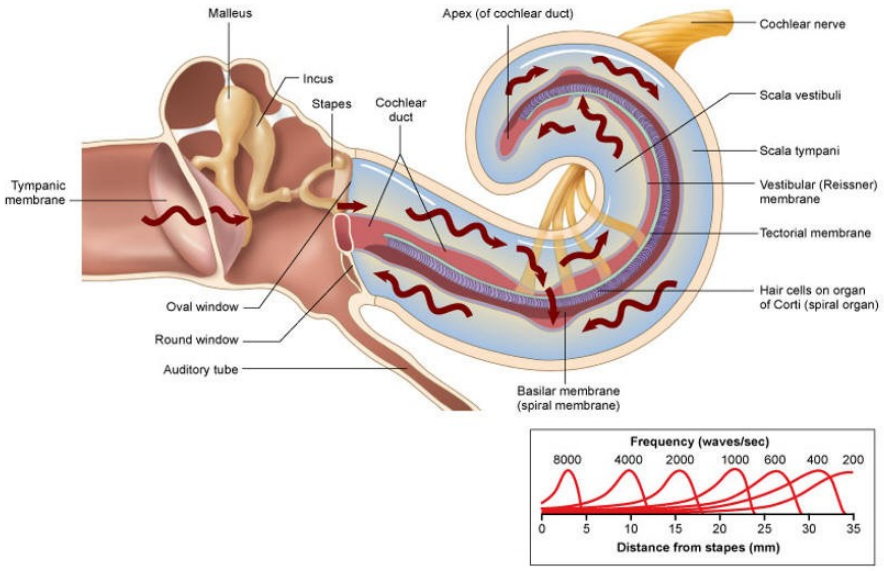


FIGURE 1.12: Effect of sound waves on cochlear structures. Image taken from (Patton et al., 2012).

altering the impulse frequency of the action potentials. These action potentials travel to the brain through the vestibulocochlear nerve (Hull, 2011).

The intensity of the auditory stimulation depends on the number of action potentials per unit of time and the number of cells that have been stimulated, while the frequency of the signal depends on which specific populations of nerve fibers are activated. There exists an association between the input sound frequency and the section of the cerebral cortex that has been stimulated. The lower the frequency of the vibration of the sound is, the closer to the apex the maximum excitation of the basilar membrane will occur. For greater frequencies, the maximum excitation will take place near the oval window. Depending on the section of the basilar membrane that oscillates with higher amplitude, the hair cells of that section will be activated in a higher proportion, stimulating subsequent afferent neurons that will produce spikes. This process originated the concept of characteristic frequency, to describe the way in which neurons in the inner ear respond with a particular low threshold for sound waves with a specific frequency, and plays an important role in the tone discrimination of a sound. If the sound wave in the input corresponds to a pure tone, a specific region of the basilar membrane with a particular characteristic frequency will oscillate with a higher amplitude. On the other hand, the further the section of the basilar membrane is from the characteristic frequency of the pure tone, the weaker the response will be. Thus, each of the sections of the basilar membrane act as an

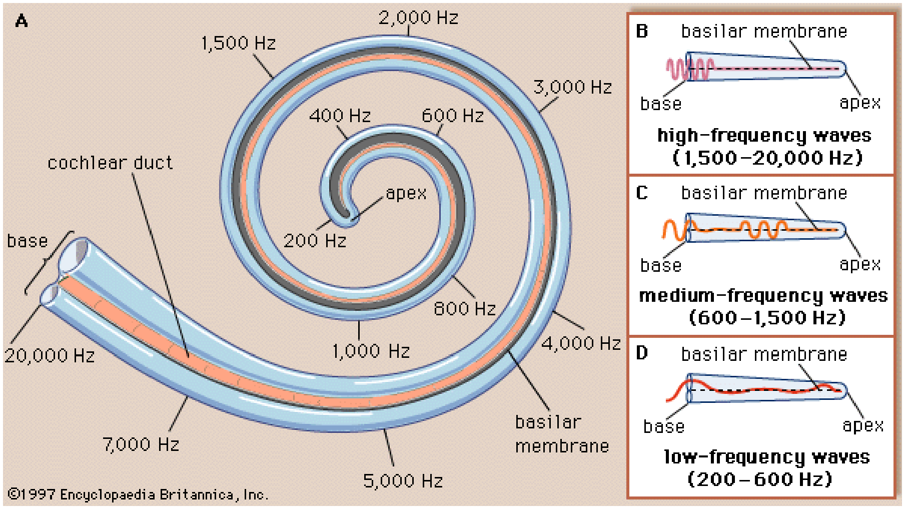


FIGURE 1.13: Tonotopic distribution of the cochlea (A). Localization of high-frequency (B), medium-frequency (C) and low-frequency (D) responses in the cochlea. Image taken from (Domínguez Morales, 2018).

auditory filter that reacts to a narrow frequency bandwidth (critical band).

A higher amplitude sound will produce a higher amplitude wave in the basilar membrane, increasing the number of IHCs that are excited, along with the number of action potentials that are generated in the afferent neurons. The difference between the IHCs and the OHCs lies on their functionality. While IHCs are in charge of transforming the amplitude of the wave into action potentials (as was explained in previous lines), the main functionality of OHCs is the ability to adapt the cochlear response depending on the input stimulus received. Thus, because of OHCs, the cochlear response is non-linear. The cerebral cortex classifies tones based on the region of the cochlea that has been excited, and their amplitudes based on the number of active neurons and their firing rate.

Summarizing, two types of signal representation can be found in the auditory nerve: the spectral representation and the temporal representation. This duality is caused by the fact that the hair cells of the cochlea, which present a tonotopic organization, generate a different response based on the amplitude of the signal and its temporal envelope. Therefore, for a pure tone with a specific frequency, it is represented in the auditory nerve by a position, based on the position of the IHCs that are excited with that frequency, and by the periodicity of the responses of the fibers that react to that stimulus (temporal representation).

1.3.1.2 Auditory ascending pathway

The central auditory nervous system consists of the auditory pathways and the auditory cortex. The signals generated in the Organ of Corti inside the cochlea are sent through the vestibulocochlear nerve to the auditory cortex. The auditory cortex is the part of the temporal lobe of the brain that processes auditory information in humans and other vertebrates.

From the vestibulocochlear nerve to the auditory cortex, the signal travels through the ascending auditory pathway (also known as the afferent pathway), where other important tracts and nuclei of the central auditory nervous system exist. Fig. 1.14 shows a highly schematic and simplified diagram with only the main components, although other nuclei exist. Almost all fibers of the auditory nerve synapse on cells of the Cochlear Nucleus (CN) (comprising the Ventral Cochlear Nucleus (VCN) and the Dorsal Cochlear Nucleus (DCN)), where the processing of the acoustic information begins. Signals can take different paths in their way from the cochlear nucleus to the auditory cortex. Most of the axons of the cochlear nucleus cells (around 70%) cross over to the opposite side (contralateral side) of the brain, whereas only 30% of them are connected with other elements in the same side of the brain as the ear from which the signal was received. Thus, each of the hemispheres of the brain cortex receives auditory information from both ears (Hull, 2011).

Both crossed and uncrossed axons of the cochlear nuclei synapse in an area of the central auditory nervous system called the Superior Olivary Complex (SOC), which is the first place in the afferent pathway of the auditory system that receives information from both ears. The superior olivary complex is divided into three main different nuclei: the Medial Superior Olive (MSO), the Lateral Superior Olive (LSO) and the Medial Nucleus of the Trapezoid Body (MNTB). The MSO is believed to measure the time difference of the arrival of sounds between the ears (Interaural Time Difference (ITD)), whereas the LSO is believed to be involved in measuring the difference in sound intensity between both ears (ILD). Both ITD and ILD are very important for determining the azimuth of sounds, i.e. localizing the input sound in the space.

Then, the impulses are transmitted to the Inferior Colliculus (IC), whose main functionalities are signal integration from various auditory nuclei, frequency recognition and pitch discrimination (Skottun et al., 2001; Shore, 2009). From there, the information goes to the Medial Geniculate Body (MGB) (also known as medial geniculate nucleus), which is part of the auditory thalamus and serves as a connection between the IC and the auditory cortex. Neurons in the auditory cortex are organized based on the frequencies of the sound to which they respond best, e.g. a frequency map (known as tonotopic map). The impulses generated by the hair cells in a particular section of the basilar membrane (sensitive to a specific frequency interval) are projected in the corresponding region in the auditory cortex. Therefore, the brain perceives the tone received

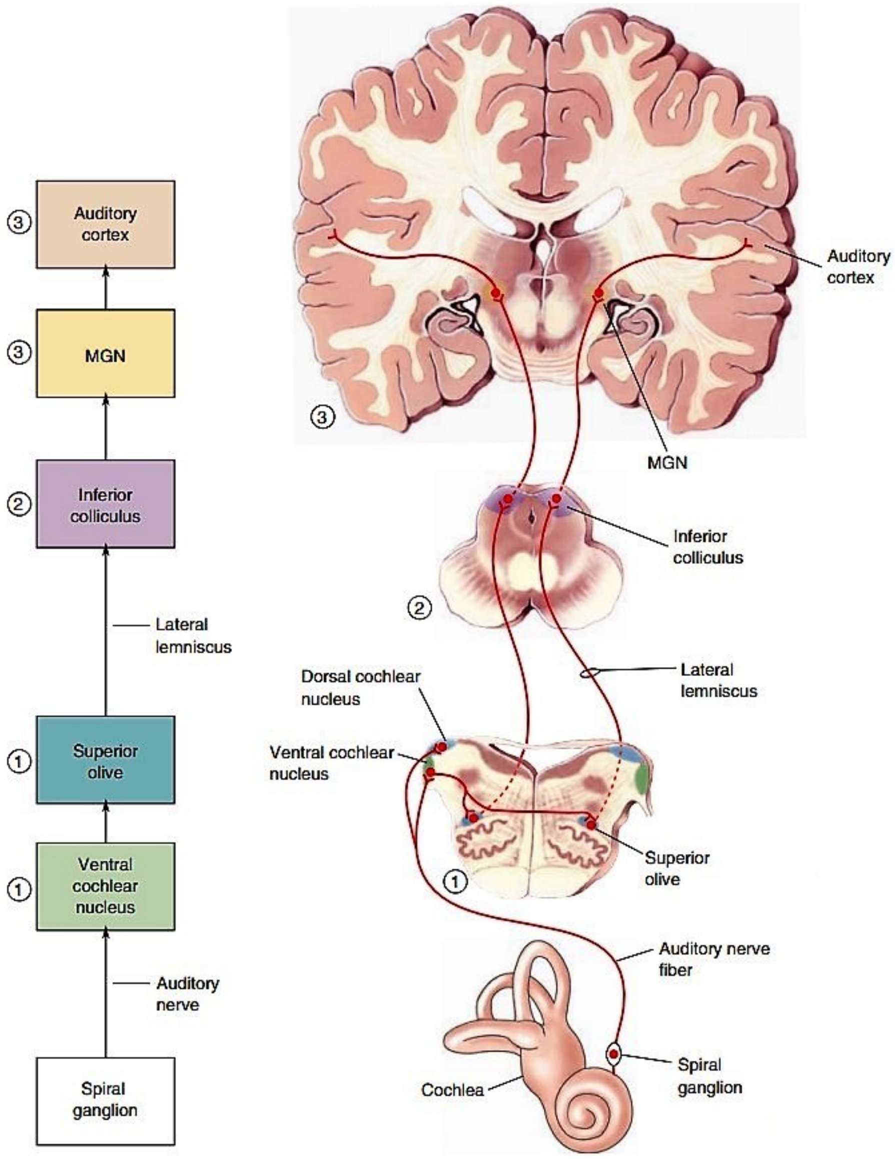


FIGURE 1.14: Ascending Auditory Pathways scheme. Neural signals can travel from the spiral ganglion to auditory cortex via numerous pathways. Here, a primary pathway is shown schematically (at left) and through brain stem cross sections. Notice that only the connections from one side are illustrated at right image. Image taken from (Bear et al., 2020).

in the ear based on the area of the cortex that is stimulated. The auditory cortex also has neurons that respond best to the action potentials generated by high amplitude sounds, while others are more sensitive to low amplitude sounds. Part of the auditory cortex is responsible for giving sense to the information, discriminating between different sound patterns based on its variations in tone, frequency, intensity and direction. The auditory cortex also receives information from other regions of the brain such as the visual cortex and the somatosensory cortex, collaborating in the interpretation of the signals (Hull, 2011).

The auditory system also transmits information from the auditory cortex to the cochlea through the efferent pathway (also known as the descending auditory pathway), giving feedback and modifying the analysis that is made in the cochlea in the form of frequency discrimination or non-linear amplification of quiet sounds. OHCs play an important role in this process (Cant and Benson, 2003).

Humans are able to perceive sounds in a frequency region between 20 Hz and 20000 Hz, although we are more sensitive to the ones between 2000 Hz and 5000 Hz. This range differs between two different persons, shrinking during life, particularly the perception of high frequency sounds (Rodríguez Valiente et al., 2014).

The frequency resolution of the ear depends on the intensity and frequency of the input sound in such a way that the lower the frequency of the input sound is, the lower the resolution will be. For tones around 200 Hz, we are able to discriminate sounds with an accuracy of 1 Hz. For higher frequencies, the frequency resolution increases. For tones around 10 kHz, the frequency resolution is 200 Hz. The reason for this is that the auditory system acts as a set of overlapped filters, in which the lower frequencies involve narrower filters and the higher frequencies involve wider filters (Janus, 2004).

Regarding the duration of the sound, it only has an inferior limit. The shortest perceptible sound range between 10 and 40 ms (Bascuas, 1997). Twenty milliseconds is a widely used value when integrating information of sounds in auditory processing.

1.3.1.2.1 Cochlear nuclei

The afferent nerve fibers from the cochlear cells are connected to the vestibulocochlear nerve, that terminate in the Cochlear Nucleus (CN). More specifically, those fibers project in a tonotopic pattern to the cochlear nuclei, distributing the projections between both the VCN and DCN. The former encodes the lower frequencies, whereas the latter encodes higher frequencies (see Fig. 1.15). In addition, each cochlear nerve fiber stimulate different areas within the CN, thus activating different types of neurons and therefore obtaining diverse response patterns. Consequently, it can be said that the auditory ascending pathway is split into pathways which extract different cues from the input acoustic information.

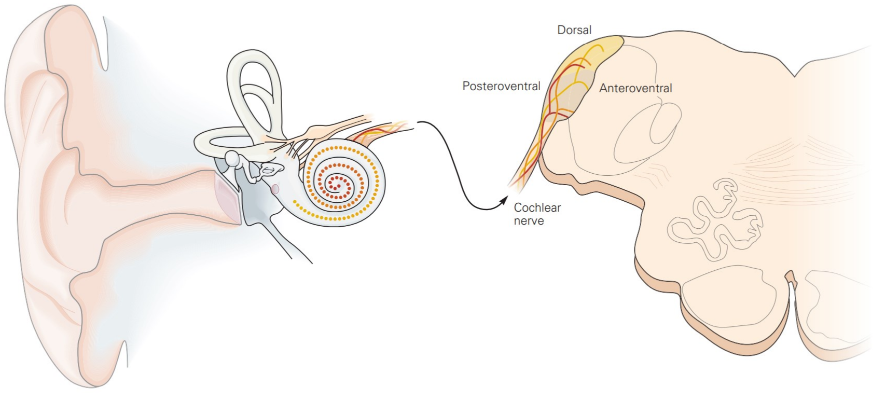


FIGURE 1.15: Cochlear nerve fibers terminate in the dorsal and ventral cochlear nuclei in a tonotopic organization. Image taken from (Kandel et al., 2000).

For instance, the cells of the VCN are able to extract timing and spectral cues that are then sent to other auditory nuclei for further processing. Among those cells, bushy cells can be considered as the one of the most important cells in this nuclei. They project bilaterally to the Superior Olivary Complex (SOC), and the projections can be split in two parts: 1) through the MSO (larger spherical bushy cells) and 2) through the LSO and MNTB (smaller spherical bushy cells). In the first case, these cells are sensitive to low frequencies, and projects in a bilateral way to the MSO. The distribution of cells in the MSO forms a circtuit that is able to detect interaural time delays, so-called ITD, thus permitting the localization of sounds in the horizontal plane (low-frequency sounds). In the second case, those cells are sensitive to high frequencies, and project the LSO ipsilaterally. In addition, globular bushy cells excite neurons in the contralateral MNTB that in turn inhibit principal cells of the LSO. The pathways through the LSO are involved in the detection of interaural intensity differences, so-called ILD, and contribute to the localization of high-frequency sounds in the horizontal plane (Kandel et al., 2000).

Stellate cells excite neurons in the ipsilateral dorsal cochlear nucleus, probably in the ipsilateral LSO, in the periolivary nuclei, and in the contralateral ventral nucleus of the lateral lemniscus through collaterals of axons that project to the contralateral Inferior Colliculus (IC). The tonotopic array of stellate cells encodes the spectra of sounds. Octopus cells excite targets in the contralateral periolivary region and the ventral nucleus of the lateral lemniscus. These neurons detect onset transients and periodicity in sounds and may be involved in the recognition of sound patterns.

Octopus and bushy cells in the VCN are able to respond with exceptionally rapid and precisely timed synaptic potentials. These neurons have a low input resistance and rapid responsiveness and prevents repetitive firing. Neurons in the VCN are able to encode different features of sounds because of differences in the pattern of input and their biophysical properties.

Octopus cells detect synchronous firing in cochlear nerve fibers with exceptional temporal precision. Individual octopus cells detect coincident firing in the relatively large number of cochlear nerve fibers (more than 60) that contact them. Coincident firing in large numbers of cochlear nerve fibers is produced by periodic sounds such as vowels and musical sounds and by the onset of broadband sounds found in consonants or clicks.

Compared to octopus cells, bushy cells convey a more sharply tuned but less temporally precise version of the firing patterns of cochlear nerve fibers. The roughly 10 cochlear nerve fibers that terminate on each bushy cell deliver relatively large synaptic currents that require summation of only a few inputs to trigger an action potential. Two properties of bushy cells enable these neurons to encode with precision the detailed temporal structure of sounds. First, the cells' low input resistance shortens the voltage changes produced by the incoming synaptic currents; their need to summate several inputs removes variability in the timing of firing in cochlear nerve fibers by averaging.

Second, the temporal fine structure of sounds that bushy cells encode provides information about the relative time of arrival of inputs to the two ears and is used at the next synaptic stage to form a map of the Interaural Time Difference (ITD) that underlie the ability to localize sound sources in the horizontal plane. The detection of musical pitch also requires the encoding of the temporal fine structure of sounds, but whether that information is carried through octopus or bushy cells or through a combination of pathways is not known (Kandel et al., 2000).

According to the studies, the main source of information about sound frequency that complements information derived from tonotopic maps is the timing of neural firing. Recordings made from neurons in the auditory nerve show phase locking, the consistent firing of a cell at the same phase of a sound wave (see Fig. 1.16). Considering a sound wave as a sinusoidal variation in air pressure, a phase-locked neuron would fire action potentials at either the peaks, the troughs, or some other constant location on the wave. At low frequencies, some neurons fire action potentials every time the sound has a particular phase (see Fig. 1.16a). This makes it easy to determine the frequency of the sound; it is the same as the frequency of the neuron's action potentials (Bear et al., 2020).

Phase locking can still occur even if an action potential has not fired on every cycle (see Fig.1.16b). For instance, a neuron may respond to a 1000 Hz sound with an action potential on only perhaps 25% of the cycles of the input, but those action potentials will always occur at the same phase of the sound. If you have

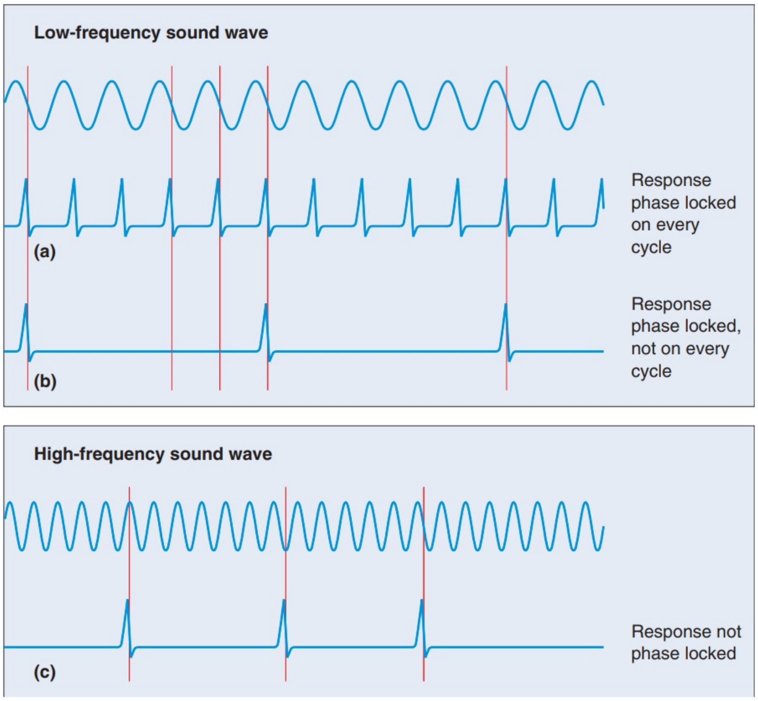

Low-frequency sound wave
Response phase locked on every cycle
(a)
Response phase locked, not on every cycle
(b)
High-frequency sound wave
Response not phase locked
(c)

FIGURE 1.16: Phase locking in the response of auditory nerve fibers. Sound at a low frequency can elicit a phase-locked response, either (a) on every cycle of the stimulus or (b) on some fraction of the cycles. (c) At high frequencies, the response does not have a fixed-phase relationship to the stimulus. Image taken from (Bear et al., 2020).

a group of such neurons, each responding to different cycles of the input signal, it is possible to have a response to every cycle (by some member of the group) and thus a measure of sound frequency. It is likely that intermediate sound frequencies are represented by the combined activity of a number of neurons, each of which fires in a phase-locked manner; this is called the volley principle. Phase locking occurs with sound waves up to about 4 kHz. Above this point, the action potentials fired by a neuron are at random phases of the sound wave (see Fig. 1.16c) because the intrinsic variability in the timing of the action potential becomes comparable to the time interval between successive cycles of the sound. In other words, the sound waves cycle too fast for the action potentials of single neurons to accurately represent their timing. Above 4 kHz, frequencies are represented by tonotopy alone.

1.3.1.2.2 Superior Olive

It is important to recall that neurons in the cochlear nuclei only receive afferents from the ipsilateral auditory nerve. Thus, all of these cells are monaural neurons, meaning that they only respond to the sound presented to one ear. However, at all later stages of processing in the auditory system, there are binaural neurons whose responses are influenced by sound at both ears. The response properties of binaural neurons imply that they play an important role in sound localization in the horizontal plane.

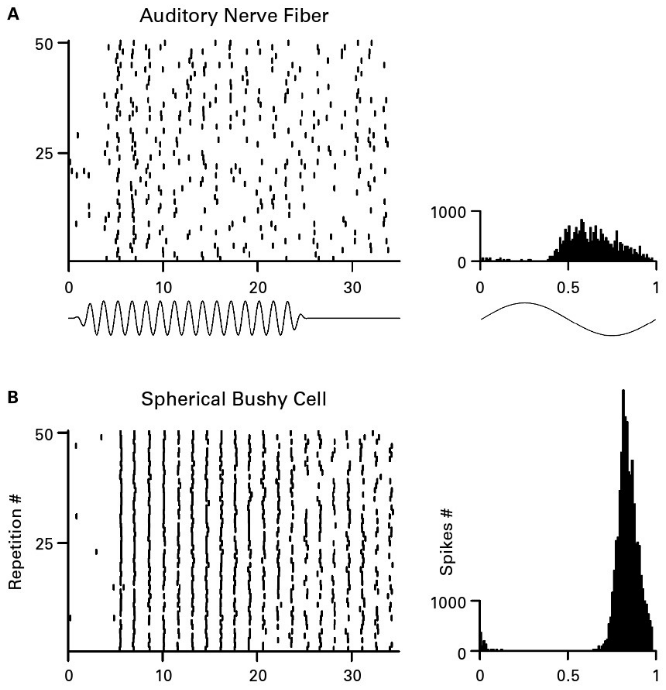


FIGURE 1.17: Phase-locked discharges of an auditory nerve fiber (A) and a spherical bushy cell of the AVCN (B). Image taken from (Schnupp et al., 2011).

The first structure where binaural neurons are present is the Superior Olive, also known as Superior Olivary Complex (SOC). Neurons in the SOC receive input from CN on both sides of the brain stem (see Fig.1.14). Cells in the cochlear nuclei that project to the superior olive typically have responses phase locked to lower-frequency sound inputs. Fig. 1.17 shows an example of non phase-locked response from the auditory nerve fiber (A) versus phase-locked responses from spherical bushy cells (B).

In many vertebrates, including mammals and birds, neurons in the SOC compare the activity of cells in the bilateral CN to locate sound sources. The SOC is a collection of brainstem nuclei that takes part in multiple aspects of hearing and is an important component of the ascending and descending auditory pathways of the auditory system. Separate circuits detect interaural time and intensity differences. The SOC is divided into three primary nuclei: the MSO, LSO, and the MNTB, and several smaller periolivary nuclei. Fig. 1.18 shows those nuclei in the SOC as well as the cochlea and the cochlear nuclei. The color code indicates the frequency sensitiveness of each nuclei and its distribution, where *HF* means "high frequency" and *LF* means "low frequency".

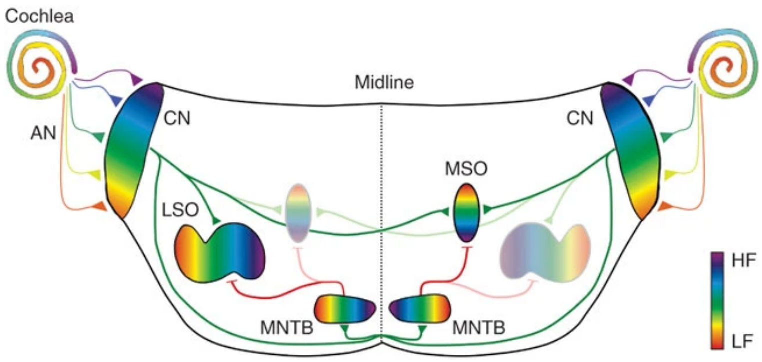


FIGURE 1.18: Schematic drawing of primary auditory sound localization circuits in the mammalian brainstem. Image taken from (Kandler et al., 2009).

The MSO is thought to help locate the azimuth of a sound, i.e., the angle to the left or right where the sound source is located. Sound elevation cues are not processed in the olivary complex since the elevation is extracted by analyzing the spectral features of the input sound. Differences in arrival times at the ears are not represented at the cochlea. Instead, a map of interaural phase is created in the MSO by comparing the timing of firing in responses to sounds from the two ears. Sounds arrive at the near ear before they arrive at the far ear, with ITDs being directly related to the location of sound sources in the horizontal plane.

The neurons in the MSO are sensitive to their own preferred ITDs. Although individual neurons may fail to fire at some cycles, the population of neurons represents the fine structure of sound waves by firing with every cycle. In so doing, these neurons carry information about the timing of inputs with every cycle of the sound. Sounds arriving from the side evoke phase-locked firing that is consistently earlier at the near ear than at the far ear, resulting in consistent ITDs. In 1948, Lloyd Jeffress suggested that an array of detectors of coincident inputs from the two ears, transmitted through delay lines comprising axons with systematically differing lengths, could form a map of ITDs and thus a map of the location of sound sources. Fig. 1.19 shows a schematic of the model proposed by Jeffress in (Jeffress, 1948).

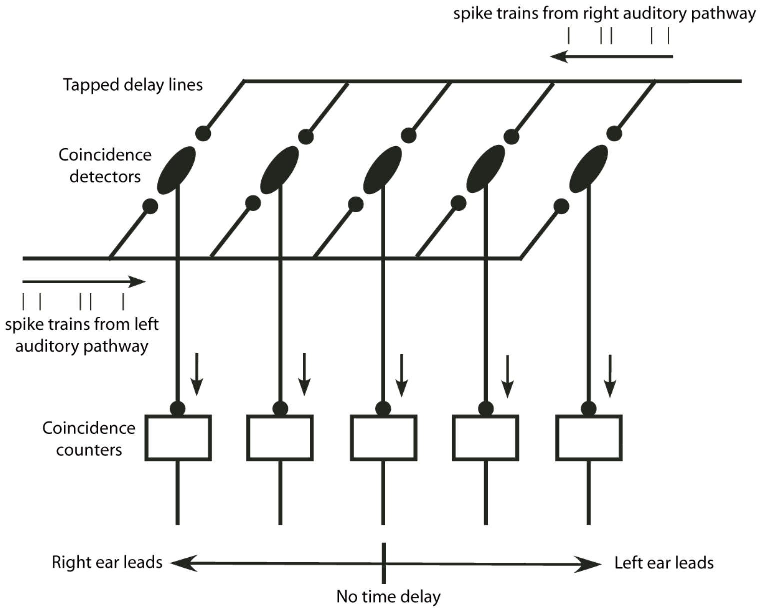


FIGURE 1.19: The tapped delay lines, coincidence detectors and coincidence counters (rate integrators) of the Jeffress model. Image taken from (Cariani, 2011).

Humans are capable of using interaural time-of-arrival differences (ITDs) of as small as 10-20 microseconds to distinguish directional differences of sound sources in the horizontal plane as small as 1-2 degrees (azimuth). Typically, ITDs range from zero for sounds coming from directly in front to about 700 microseconds for sounds coming directly from either side. Traveling around the head takes about 700 μs, and the medial superior olive is able to distinguish time differences much smaller than this. In fact, it is observed that people can detect interaural differences down to 10 μs.

However, this mechanism does not work with continuous tones at high frequencies. If a sound coming from the right has a frequency of 20000 Hz, it means that one cycle of the sound covers 1.7 cm. After a peak reaches the right ear, it would take less than 0.6 μs before a peak arrives at the left ear, thus receiving many peaks of such a high-frequency wave in that time period. Therefore, no longer is there a simple relationship between the direction the sound comes from and the arrival times of the peaks at the two ears. Interaural arrival time is not that useful for locating continuous sounds with frequencies so high that one cycle of the sound wave is smaller than the distance between your ears (i.e., greater than about 2000 Hz) (Bear et al., 2020).

Fortunately, the brain has another process for sound localization at high frequencies. An Interaural Level Difference (ILD) exists between the two ears because your head effectively casts a sound shadow (Bear et al., 2020). There is a direct relationship between the direction the sound comes from and the extent to which your head shadows the sound to one ear. If sound comes directly from the right, the left ear will hear a significantly lower intensity. With sound coming from straight ahead, the same intensity reaches the two ears, and with sound coming from intermediate directions, there are intermediate intensity differences.

Neurons sensitive to differences in intensity can use this information to locate the sound. Intensity information cannot be used to locate sounds at lower frequencies because sound waves at these frequencies diffract around the head, and the intensities at the two ears are roughly equivalent. Therefore, there is no sound shadow at low frequencies.

Although the LSO does not form a map of the location of sounds in the horizontal plane, it performs the first of several integrative steps that use ILDs to localize sounds. Neurons in this nucleus balance excitatory input from small spherical bushy cells and stellate cells in the ipsilateral VCN with inhibitory input from a disynaptic pathway that includes globular bushy cells in the contralateral VCN and principal neurons of the ipsilateral MNTB. Sounds that arise ipsilaterally generate relatively strong excitation and relatively weak inhibition whereas those that arise contralaterally generate stronger inhibition than excitation. Differences in the location of the sound sources that generate differences in the balance between excitation and inhibition are reflected in the firing rates of neurons in the LSO. Neurons in the LSO are thus activated more strongly by sounds from the ipsilateral than from the contralateral hemifield (Kandel et al., 2000).

Summarizing the two processes for localizing sound in the horizontal plane: with sounds in the range of 20–2000 Hz, the process involves interaural time delay. From 2000–20,000 Hz, interaural intensity difference is used. Together, these two processes constitute the duplex theory of sound localization.

1.3.1.2.3 Inferior colliculus

The Inferior Colliculus (IC) occupies a central position in the auditory pathway of all vertebrate animals because all auditory pathways ascending through the brain stem converge there 1.14. Cells in the VCN send out axons that project to the SOC on both sides of the brain stem. Axons of the olivary neurons ascend in the lateral lemniscus (a lemniscus is a collection of axons) and innervate the IC of the midbrain. Many efferents of the dorsal cochlear nucleus follow a route similar to the pathway from the VCN, but the dorsal path bypasses the superior olive. Although there are other routes from the cochlear nucleus to the IC, with additional intermediate relays, all ascending auditory pathways converge onto the IC. The neurons in the IC send out axons to the

Medial Geniculate Nucleus (MGN) of the thalamus, which in turn projects to auditory cortex.

Many neurons in the central nucleus carry information about the location of sound sources. The majority of these cells are sensitive to interaural time and intensity differences, which are known to be essential cues for localizing sounds in the azimuth. Neurons are also sensitive to spectral cues that localize sounds in the vertical plane. To localize sounds accurately, animals must ignore the re1ections of sounds from surrounding surfaces that arrive after the initial direct wave front.

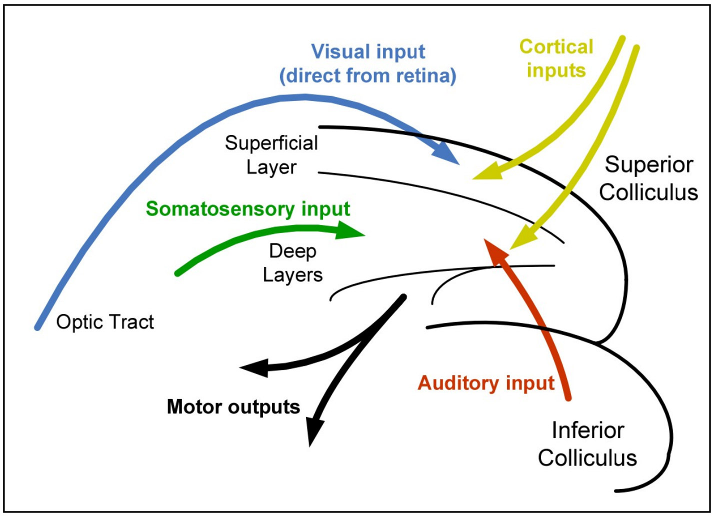


FIGURE 1.20: Sensory and cortical information flows in the SC. The superficial layer of the SC is directly connected to the optic tract, while the deep layers receive auditory (from the inferior colliculus) and somatosensory input. Cortical feedback moderates the processing in the deep layer and influences the motor outputs. Image taken from (Pavlou and Casey, 2010).

It is also important to mention that the IC sends axons not only to the MGN but also to the Superior Colliculus (SC), where the integration of auditory and visual information occurs, and to the cerebellum, as shown in Fig. 1.20. In addition, there is extensive feedback in the auditory pathways. For instance, brain stem neurons send axons that contact outer hair cells, and auditory cortex sends axons to the acMGN and IC.

The SC is critical for reflexive orienting movements of the head and eyes to acoustic and visual cues in space. By the time they reach the SC, binaural sound cues and the monaural spectral cues that underlie mammalian sound localization merge to create a spatial map of sound, an auditory map, in which neurons are unambiguously tuned to specific sound directions. This convergence is critical, for the binaural level and timing differences alone cannot unambiguously code for a single position in space. The spectral cues that provide information about vertical location are essential. Different locations in the vertical plane can give rise to identical interaural time or intensity differences. Such a spatial map is formed both in birds and in some mammals (Kandel et al., 2000). Within the SC the auditory map is congruent with maps of visual space and the body surface. Unlike the visual and somatosensory maps, the auditory map is computed from a combination of cues that identify the specific position of a sound source in space, and is not based on the peripheral receptor surface.

Auditory, visual, and somatosensory neurons in the SC all converge on output pathways in the same structure that controls orienting movements of the eyes, head, and external ears. The motor circuits of the SC are mapped with respect to motor targets in space, and are aligned with the sensory maps. Such sensory-motor correspondence facilitates the sensory guiding of movements (Naveros et al., 2019).

1.3.2 Implementations of the auditory system on circuits

Artificial cochleae model the basilar membrane using a set of filters or resonators with cutoff/center frequencies (depending on whether low-pass filters or band-pass filters are implemented), which mimic the frequency distribution along the basilar membrane. Depending on the position of the basilar membrane, its flexibility and width changes. These changes are implemented by setting different parameters to the filters.

As has been presented in previous sections, the biological cochlea is a very complex part of the inner ear, which means that modelling and implementing its functionality is not an easy task. This fact has resulted on artificial cochleae implementations that only model part of the characteristics of the biological cochlea. These characteristics are chosen depending on the application that researchers want to integrate in the system. Therefore, the majority of artificial cochleae do not achieve comparable results to the ones obtained by a biological cochlea.

Since the first design of an artificial cochlea presented by Richard Lyon and Carver Mead in 1988 (Lyon and Mead, 1988), activities referred to the implementation of different mathematical models of the cochlea using analog VLSI have increased. Digital implementations using reconfigurable logic devices have also increased in popularity, although not as much as analog approaches.

Even three decades after the first silicon cochlea, artificial cochleae are still far from being comparable to the biological cochlea, especially in terms of power consumption, frequency range, dynamic range of the input, or noise immunity. Taking into account that these models aim to emulate in the same way as a system that has evolved for hundreds of millions of years, some good approximations have been achieved.

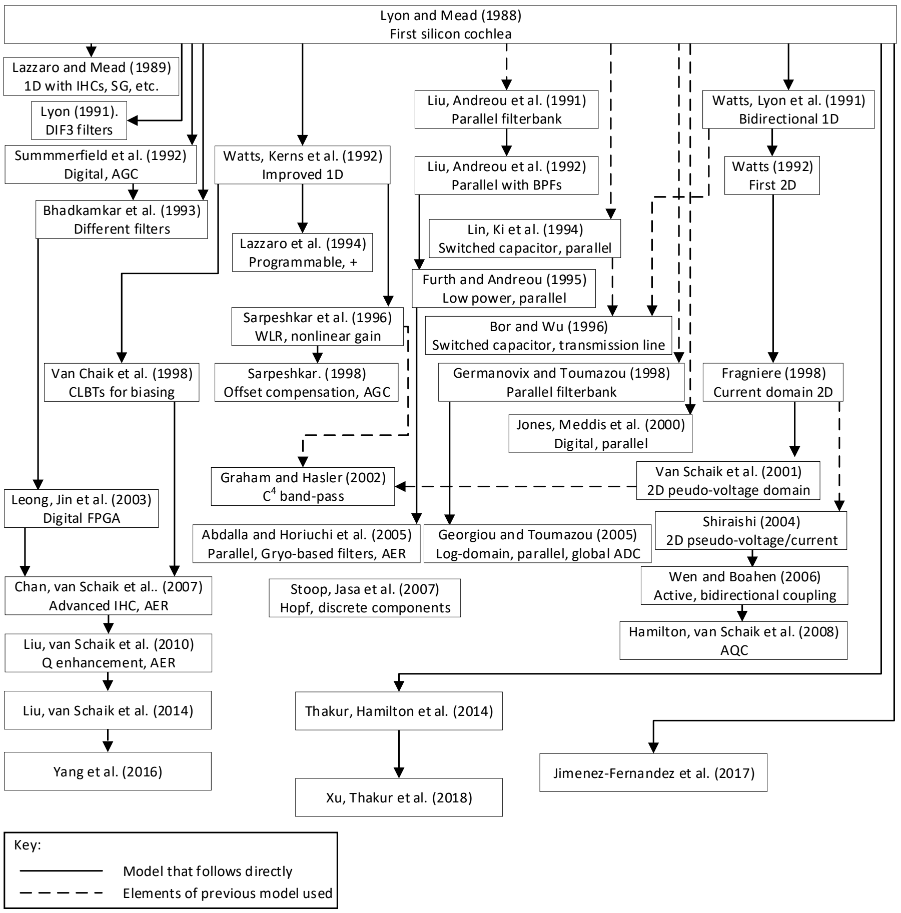


FIGURE 1.21: Historical tree diagram of different artificial cochleae developed. Image taken from (Domínguez Morales, 2018)

Fig. 1.21 shows an scheme of different implementations of artificial cochleae and their evolution in time.

Artificial cochleae can be categorized in many ways. Generally, they are classified based on:

- The coupling coefficient that exists between the elements of the filters. We distinguish between one-dimensional silicon cochlea (1-D) and two-dimensional silicon cochlea (2-D).

- The existence or lack of automatic gain control to allow filters to dynamically adapt to the intensity changes in the input (active or passive cochleae).

In this section, the most relevant developments regarding artificial cochleae are presented, both analog and digital implementations. Each of them is based on mathematical models that represent the propagation of the sound wave through the inner ear and the conversion between acoustic energy and nerve impulses for later processing. Some of the most popular mathematical models of the auditory system are explained in this section before describing analog and digital cochleae implementations.

1.3.2.1 Artificial cochleae

As it has been detailed in previous sections, part of the human auditory system acts as a continuous and overlapped set of band pass filters, which correspond to a specific region of the basilar membrane. Based on the structure of the band pass filters that are implemented, two different auditory models can be found: parallel (independent filters that process the input information at the same time) or cascade (a set of filters that are connected, where the input of one of them is the output from the previous one in the cascade).

In the response of the critical bands of the basilar membrane, a very pronounced peak can be observed, which corresponds to the resonant frequency, and an attenuation for frequencies above or below the resonant frequency, with the slope being steeper for higher frequencies. This effect can be achieved with a cascade band pass filter. A parallel band pass filter would need higher order filters to implement that behavior.

1.3.2.1.1 Lyon's model

A cochlear model based on the knowledge of the biological cochlea and its main functionalities was developed by Richard F. Lyon (Lyon, 1982). This model describes the propagation of sound in the inner ear and the conversion of acoustic energy into neural representations. When sound reaches the cochlea, a wave travels through the basilar membrane. The physical properties of the basilar membrane change from the base where the oval window is to the apex, so that the frequency components of the wave reach a maximum in a particular position of the basilar membrane.

The cochlear model described by Lyon combines a set of filters to represent the waves that travel through the basilar membrane, Half-Wave Rectifiers (HWR), to detect the energy of the signal, acting as IHCs, and different Automatic Gain Control (AGC) stages, to model the behavior of the OHCs. Due to the fact that the fundamental frequency of the basilar membrane decays exponentially from the base to the apex, the basilar membrane is divided into sections with the same length to obtain the frequency distribution of the auditory filters.

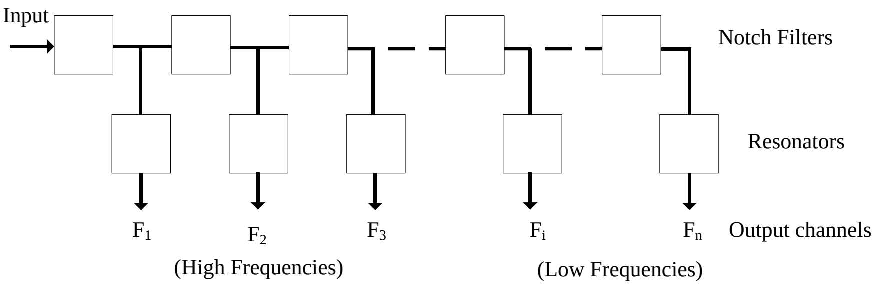


FIGURE 1.22: Lyon's model's cochlea's filters' block diagram. Image taken from (Domínguez Morales, 2018)

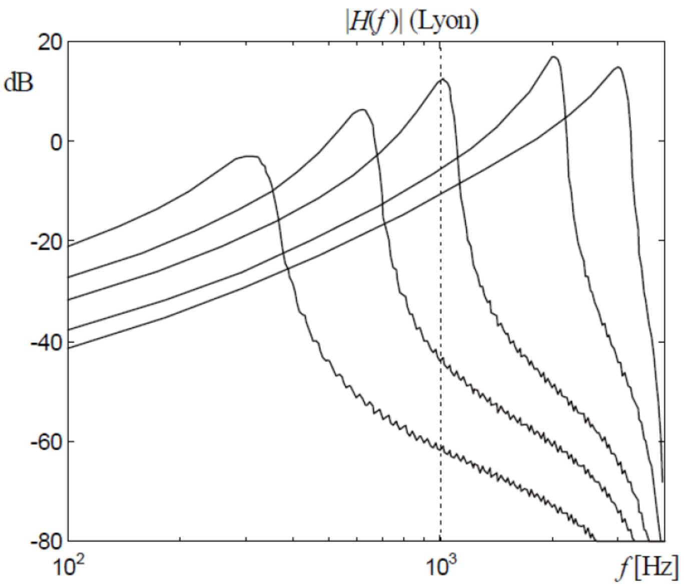


FIGURE 1.23: Frequency response of Lyon's model (64 sections) for the following characteristic frequencies: 3.0, 2.0, 1.0, 0.6 and 0.3 kHz. Image taken from (Miró Amarante, 2013)

At each point in the cochlea, the acoustic wave is filtered by a notch filter[4]. Each notch filter operates at successfully lower frequencies so the net effect is to gradually low-pass filter the acoustic energy. An additional resonator (or

[4]The notch filter, also known as band-stop filter or band-rejection filter, is a filter that passes most frequencies unaltered, but attenuates those in a specific range to very low levels.

bandpass filter) picks out a small range of the traveling energy and models the conversion into basilar membrane motion. It is this motion of the basilar membrane that is detected by the inner hair cells. The block diagram that represents this cascade architecture is shown in Fig. 1.22. This way, the high frequency components of the signal are filtered while the remaining components of the signal travel through the cascade of filters. The rejection of the high frequency components produces a steep slope in the frequency response of the filters that correspond to low frequencies. The biological cochlea also presents this behavior. Therefore, this model provides a good approximation to the processing that is done in the cochlea.

Fig. 1.23 presents the response of the Lyon's model for a configuration with 64 sections and a sampling frequency of 8 kHz. It is important to highlight the difference between the ascending and the descending slopes with respect to the fundamental frequency. The higher the fundamental frequency, the higher the slope immediately after. This effect is due to the rejection of the high frequency components across the cascade of filters. The attenuation of the signal at low frequencies is caused by the inclusion of a preemphasis filter, which roughly models the effects of the outer and middle ear.

1.3.2.1.2 Lyon & Katsiamis' model

Richard Lyon, together with Andreas Katsiamis and Emmanuel Drakakis, published a research paper in 2007, in which they presented transfer functions in the continuous domain that were designed based on gammatone filters for auditory information processing (Katsiamis et al., 2007).

In the paper, the design of two different type of filters is presented: the Differentiated All-Pole Gammatone Filter (DAPGF) and the One-Zero Gammatone Filter (OZGF). These designs are characterized for having a hardware-implementation oriented architecture; they have the same properties and functionalities as the cochlea operation and overcome some limitations of the gammatone filters, such as the symmetric response and the complexity in the frequency domain (Miró Amarante, 2013).

Unlike the cascade architecture of Lyon's model, this model is based on a bank of filters with parallel stages that consist of blocks connected in a cascade fashion. Fig. 1.24 shows how the basilar membrane can be modeled with both a parallel or a cascade architecture.

1.3.2.1.3 Analog cochleae

A one-dimensional cascade cochlea models the propagation of the sound wave through the basilar membrane in one unique direction (from the base

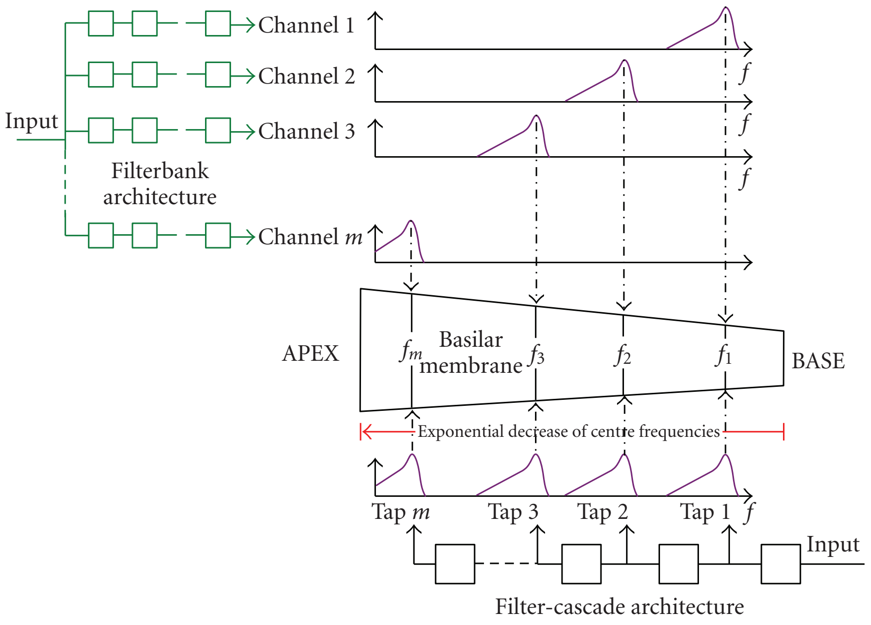


FIGURE 1.24: Graphical representation of the filterbank and filter-cascade architectures in the Lyon-Katsiamis model. Image taken from (Katsiamis et al., 2007)

of the basilar membrane to the apex). Moreover, in this cascade architecture, each section of the cochlea (auditory filter) processes the output of the previous element. Even when using second-order filters, this allows a steep slope effect to occur, which favors the frequency selectivity of the basilar membrane itself.

The first bio-inspired artificial cochlea was developed by Lyon and Mead in 1988 (Lyon and Mead, 1988) following this one-dimensional cascade topology, the Lyon's model, which was presented in section 1.3.2.1.1. (Lyon, 1982). It has been proved that, in the human cochlea, the resonant frequency across the basilar membrane decreases exponentially in a logarithmic scale: high frequencies near the base and low frequencies near the apex. To implement this artificial cochlea, the basilar membrane has been divided into different segments with the same length. A set of filters with a specific resonant frequency according to the resonant frequency of each segment of the basilar membrane is used in a cascade topology. Each of the filters have the same architecture, only changing the resonant frequency and their low-pass response. High frequency components are removed in the output of each filter, which produces a steep slope in the frequency response curves. This slope is also observed in the biological cochlea when the wave travels across the basilar membrane. Therefore, despite the simplicity of this model, it provides a first approximation of signal processing inside a biological cochlea.

The work developed by Lyon and Mead consists of 480 sections of second-order low-pass filters in a cascade topology with resonant frequencies logarithmically distributed in order to model the propagation of the wave and the frequency analysis associated to the basilar membrane. This cochlea implementation successfully modeled some particular characteristics of the biological cochlea, and has provided a starting point for the research of neuromorphic cochleae. There exist several implementations based on this first one proposed by Lyon and Mead. John Lazzaro added circuits to the Lyon and Mead's cochlea in order to model the behavior of the IHCs. These circuits encode the output spikes of the artificial cochlea (Lazzaro and Mead, 1989a).

Another implementation based on this approach is the work by Lloyd Watts (Watts et al., 1992), which achieves a better exponential distribution of the resonant frequencies. This increments the linear range and eliminates the instability of the signal.

In 1995, Lazzaro and Wawrzynek (Lazzaro and Wawrzynek, 1995) proposed an improved version of the Watts' model, adding the AER communication protocol in the output of the cochlea. It is the first artificial cochlea to use this kind of communication, which currently is widely extended in neuromorphic developments.

In 1996, Andre van Schaik, Eric Fragniere, and Eric Vittoz presented a new artificial cochlea implementation based on Watts' (Van Schaik et al., 1996). Fig. 1.25 shows the cutoff frequencies distribution of this implementation compared to the cutoff frequencies distribution achieved in Watts' work. As can be observed, in this implementation the exponential distribution of the frequencies is more uniform. Thanks to this improvement, this artificial cochlea allows binaural sounds processing, as is presented in works like (Chan et al., 2007) and (Yu et al., 2009), in which two van Schaik's artificial cochleae with a more advanced module to manage the output spikes by using the AER protocol are used for echolocation tasks. The artificial cochlea that was presented in (Chan et al., 2007) has also been used to build a classification system in order to distinguish between two sounds: a clap and a bass drum (Jäckel et al., 2010). This cochlea was improved in a number of channels, output event rate and frequency range in (Liu et al., 2010b) and (Liu et al., 2014).

The main downside of the cascade topology is the low fault-tolerance. If any of the elements of the cascade fails, this error will be propagated to the rest of the subsequent elements. Moreover, it is important to notice that each segment of the cascade adds a delay, which is inversely proportional to the center frequency of the filter. Then, the filters corresponding to low frequencies will introduce more delay than the ones corresponding to high frequencies. Therefore, the number of sections (stages or channels) limits the frequency response of the system. The noise that is generated by the filters is accumulated across the cascade, which

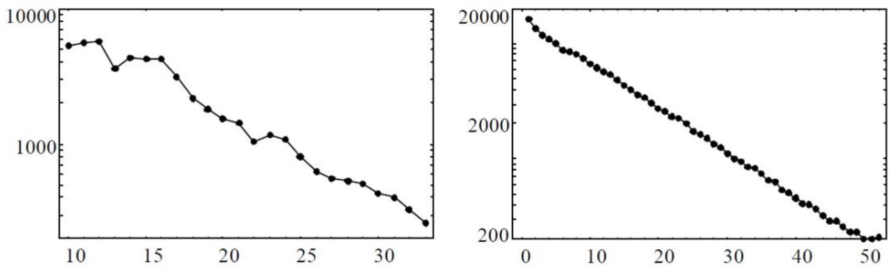


FIGURE 1.25: Cutoff frequencies (Hz) distribution in Watts' artificial cochlea (left) and van Schaik's (right). Image taken from (Van Schaik et al., 1996)

reduces the dynamic range of the system. Some of these issues can be solved with a parallel topology or a 2-D topology.

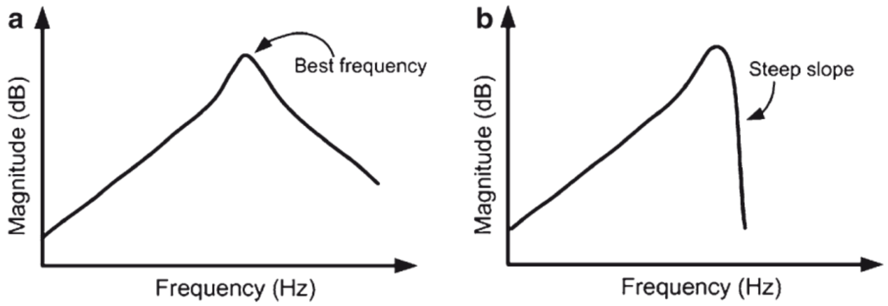


FIGURE 1.26: Frequency response of a second order filter in a parallel topology (a) and in a cascade topology (b). Image taken from (Domínguez Morales, 2018)

Parallel topologies are usually chosen for their ease of implementation. However, even though the issues of cascade topologies are not present in these models, parallel topologies are not the best option when implementing an analog cochlea because each filter acts independently from the rest and higher order filters are needed to produce the same "steep slope" effect in the high frequencies as in cascade topologies, which increments the power consumption of the system, requiring a larger silicon area. Fig. 1.26 presents a comparison between the outputs of a single second-order filter in a 1-D parallel topology (Fig. 1.26 (a)), and in a 1-D cascade topology (Fig. 1.26 (b)).

The implementations presented in (Liu et al., 1991) and (Liu et al., 1992) are some examples of parallel topologies. Following the same architecture that was implemented in these works, in 1998 a new artificial cochlea was developed for pattern extraction in speech recognition tasks (Kumar et al., 1998). In 2016, Minhao Yang, Chen-Han Chien, Tobias Delbruck and Shih-Chii Liu presented a 64-channel binaural cochlea with parallel asynchronous event output (Yang et al., 2016). In this model, each binaural channel performs feature extraction by analog bandpass filtering and, after that, filtered signals are encoded into events via Asynchronous Delta Modulation (ADM).

Bidimensional (2-D) cochleae model both the propagation of the wave across the basilar membrane and the fluid motion within the cochlea and the basilar membrane. The structure of the filters in this type of artificial cochlea are based on Lyon and Katsiamis' model, which is detailed in section 1.3.2.1.2. Neighbor filters are connected through a set of resistors that model the inner fluid of the cochlea. This architecture combines the benefits of the previous 1-D architectures presented before: the coupling of the filters in parallel allows to generate a steep slope in high frequencies despite being second-order filters; the fault tolerance is improved and the delay accumulation is avoided. In 1992, Watts presented a 50 stages (channels) implementation that improved the dynamic range, stability and errors caused by the transistors (Watts et al., 1992), achieving a uniform frequency response and good values in the quality factors of the filters (Q).

(Fragnière, 1998) describes a 2-D cochlear model in which the pressure and voltage are mathematically analogous to the acceleration of the basilar membrane and the action potentials. Some authors have published recent implementations of this model (Van Schaik and Fragnière, 2001; Shiraishi, 2003; Hamilton et al., 2008; Wen and Boahen, 2009).

The artificial cochleae implemented in the last two works mentioned before (Hamilton et al., 2008; Wen and Boahen, 2009), are active. An artificial cochlea is active when filters change dynamically depending on the input signal. Generally, the gain is increased for low frequencies and decreased for high frequencies. This behavior aims to model the functionality of the OHCs. These implementations use AGC to change the gain of the cochlea depending on the changes produced in the input signal. However, in these cases, the AGC also controls the quality factor (Q) of each section of the cochlea. This way, the gain and the bandwidth of each filter are changed dynamically based on the changes in the characteristics of the input sound.

1.3.2.1.4 Digital cochleae

The need for more efficient systems with lesser power consumption and lesser cost led to the research and development of new digital design techniques and architectures. FPGAs are a very good option for the development of neuromorphic systems. They are flexible, robust to temperature changes, they have a better dynamic range, a better Signal-to-Noise Ratio (SNR)[5], a simpler

[5]SNR, or Signal-to-Noise Ratio, is defined as the ratio of signal power to noise power, often expressed in decibels (dB).

computer interface, the board can be used for different applications and their development time is much shorter compared to analog VLSI systems. However, they move away from the biological models.

The first digital cochlea was implemented in 1992 using an ASIC (Summerfield and Lyon, 1992), Fig. 1.27. It contains 71 sections of cascade filters based on Lyon's cochlear model. The output of each filter is connected to a HWR that mimics the functionality of OHCs along with an AGC. Therefore, even though the model in which it is based is not an active cochlea (it does not have automatic gain control), this digital implementation includes it.

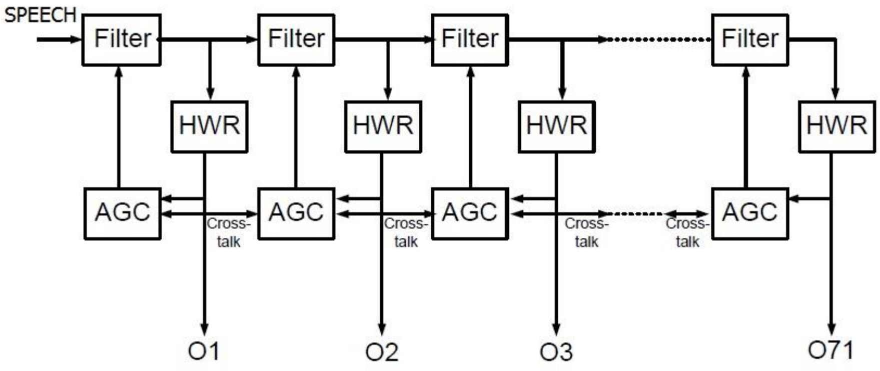


FIGURE 1.27: Block diagram of the digital cochlea proposed by Summerfield et al. Image taken from (Summerfield and Lyon, 1992)

The digital cochlea of (Jones et al., 2000) implemented a second-order band-pass filter bank with 30 filters in an FPGA. This filter bank follows a parallel architecture and it was specifically built for extracting the tone out of complex sounds. Although this digital cochlea has a simple architecture, it implements the model of IHC and auditory pathways in detail.

In (Leong et al., 2003), an 88-sections cascade was implemented by using second order Infinite Impulse Response (IIR) filters. This kind of filters are able to obtain higher and narrower bands with less arithmetic operations. Fig. 1.28 shows the frequency response of some of the sections of the cochlea presented in that work.

Parallel architectures achieve a higher speed in the output, but they have a limitation in terms of the cochlea size because the resources needed increase linearly with the number of stages in order to produce the steep slope behavior in the filters. However, in cascade architectures, the design limits are determined by the speed in the output, due to the fact that it decreases linearly with the number of stages.

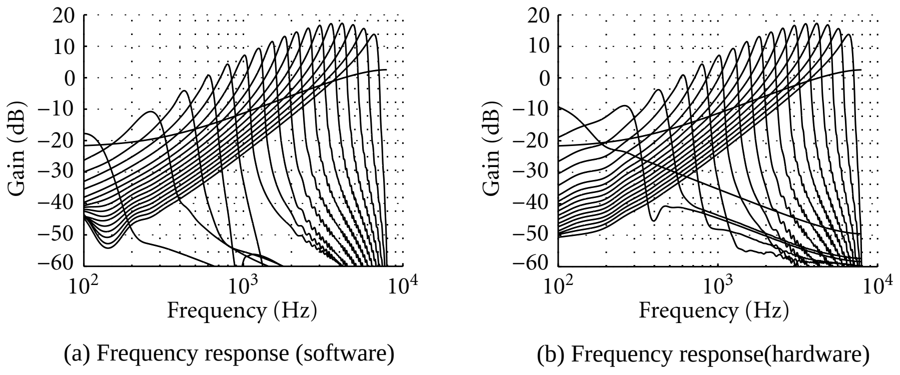


FIGURE 1.28: Frequency response of the digital cochlea implementation by Leong et al. Image taken from. Image taken from (Leong et al., 2003)

Some recent implementations of digital cochleae use IIR filters, and both parallel (Dundur et al., 2008; Miró Amarante, 2013) and cascade (Gambin et al., 2010; Mugliette et al., 2011; Thakur et al., 2014) implementations can be found. The digital design presented in (Dundur et al., 2008) is the basis for the cochlear implant implementations in FPGAs.

Other designs (Gambin et al., 2010; Mugliette et al., 2011; Thakur et al., 2014) use time multiplexing to implement the filter bank with a greater number of sections. For example, the digital cochlea presented in (Mugliette et al., 2011) has a cascade of 24 second-order IIR filters, and thanks to the time multiplexing, it only requires 20 multiplier units. Designs that are based on time multiplexing use less resources in exchange of using higher clock frequencies, increasing the power consumption of the system.

Using FPGA implementations have some advantages over VLSI analog systems: a faster design time, more robustness to power supply changes, to temperatures and to transistor mismatch, a higher dynamic range, a higher SNR, better stability, FPGAs can be reused for different applications and they have a simpler interface to the computer.

Two of the most recent digital cochlea implementations in FPGA are (Jimenez-Fernandez et al., 2017) and (Xu et al., 2018a). In (Xu et al., 2018a), a digital implementation of a 70-section, 44.1 kHz sampling rate Cascade of Asymmetric Resonators with Fast-Acting Compression (CAR-FAC) (Lyon, 2017) cochlear model is presented, where the Cascade of Asymmetric Resonators (CAR) part simulates the basilar membrane's response to sound and the Fast-Acting Compression (FAC) part models the OHCs, the IHCs, and the medial olivocochlear efferent system functions. The FAC feeds back to the CAR by moving the poles and zeros of the CAR resonators automatically, making it an active cochlea model.

The digital implementation proposed in (Jimenez-Fernandez et al., 2017), which is called Neuromorphic Auditory Sensor (NAS), is the one that has been used throughout this thesis. NAS (Jimenez-Fernandez et al., 2017) is an audio sensor for FPGAs inspired by Lyon's model of the biological cochlea (Lyon and Mead, 1988). This sensor is able to process an excitatory audio signal using Spike Signal Processing (SSP) techniques (Jimenez-Fernandez et al., 2010), decomposing incoming audio in its frequency components, and providing this information as a stream of events using the AER (Boahen, 2000). As it is implemented on a reconfigurable platform, this sensor's configuration parameters are flexible and can be adapted to any application.

NAS decomposes two digitized signals (two in case of a binaural or stereo NAS, or one in case of a monaural or mono NAS) into a set of sections or bands (corresponding to particular frequencies), which are previously converted to a spike train. The decomposition of the signals in each corresponding frequency band is performed by a bank of Spike-based Low-Pass Filter (SLPF) filters (Jimenez-Fernandez et al., 2010; Domínguez-Morales et al., 2011) connected in a cascade topology. The output of this sensor is encoded using the AER protocol as many of the neuromorphic sensors that were presented in the previous section.

For this decomposition of the input signals (left and right ear) the same processing is performed. The processing is modeled using a Spike-based Cascade Filter Bank (SCFB). Each SCFB has many stages (as many as sections in the cochlea) and consists of a SLPF and a Spike Hold&Fire (SHF) (Jimenez-Fernandez et al., 2010), which are able to extract two different spike trains: one that is passed as input to the next stage and another one that corresponds to the signal filtered by the current stage of the cascade.

This kind of sensor mimics how the biological cochlea processes audio signals. NAS is able to decompose the input audio signal into different frequency bands (also called channels). Like in other neuromorphic cochleae implementations based on a cascade model (Liu et al., 2010b; Leong et al., 2003; Summerfield and Lyon, 1992; Lyon and Mead, 1988), this decomposition is carried out by a series of cascade-connected stages that subtract the information from consecutive spike-based low-pass filters' output spikes in order to reject out-of-band frequencies, obtaining a response equivalent to that of a bandpass filter (Jimenez-Fernandez et al., 2017).

Fig. 1.29 (a) shows the global architecture of a binaural NAS. In this architecture, the first element in the chain is an AC97 audio codec (CS5344 audio codec), which has two analog signals as inputs (left and right) that are digitized and multiplexed into a single output. This signal is divided in order to obtain sampled and digitized values that correspond to each of the input signals. After that, the information is converted from a digital domain to a spike domain by using a Reverse Bitwise Synthetic Spikes Generator (RB-SSG) (Jimenez-Fernandez et al., 2010), which provides a spike stream with a frequency

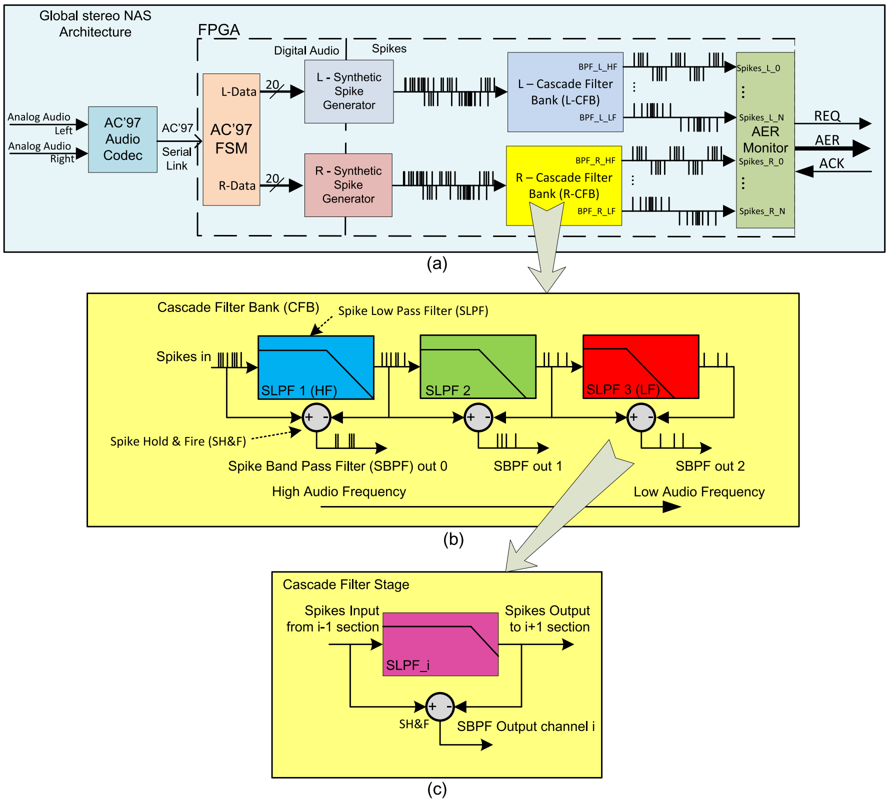


FIGURE 1.29: (a) Global NAS architecture. (b) Filter banks with Cascade topology, CFB. (c) Single CFB stage containing an SLPF and an SH&F. Image taken from. Image taken from (Jimenez-Fernandez et al., 2017)

that is proportional to the digital amplitude. At the output spike generator, ON and OFF spikes that encode the previously sampled digital value are obtained.

The generated spike train is the input to the cascaded spike-based filter bank Cascade Filter Bank (CFB) (depending on the NAS architecture implemented on the FPGA, the number of filters will be different). Fig. 1.29 (b) shows how these filters are distributed, forming the bank, where the first filters of the cascade correspond to higher frequencies, and the last ones correspond to lower frequencies. As was previously mentioned, the filter bank is divided into different stages, where each stage consists of a spiking low-pass filter along with a SHF (see Fig. 1.29 (c)). The output of each of the stages is connected to an AER monitor (Cerezuela-Escudero et al., 2013) that encodes the activity of each stage, placing the address of the corresponding stage in the output asynchronous AER bus every time that an event is generated in it.

Chapter 2

Objectives

"I have no special talent.
I am only passionately curious."
– Albert Einstein

In this chapter, the main objectives to be achieved in this thesis are listed. These objectives are divided in general objectives, presenting a global idea of the thesis; and specific objectives, where the corresponding tasks to achieve the general objectives are detailed. Then, the organization of this thesis is presented by giving a brief description of each chapter and appendix.

2.1 Objectives

According to the proposed improvements and contributions mentioned in the motivation section 1.1, two types of objectives were proposed: general objectives, with the aim of analyzing and studying the viability of a fully digital, event-based auditory system model and its implementation in both ASIC and FPGAs, for being integrated afterwards within robotic platforms; and more specific objectives, focused on solving particular problems related to the topic introduced previously and the design of the corresponding systems and models.

General objectives:

1. Develop a set of tools to provide the community to have access to a neuromorphic auditory sensor through an open-source project.
2. Study how the human auditory system captures, analyzes and encodes binaural information into nerve impulses that are processed by the brain when carrying out sound source localization tasks.
3. Integrate the auditory sensor in both general and specific-purpose robots for performing audio processing, as well as sensory integration, in real-time.

With these general objectives, it is intended to mimic the human nervous system to design neuromorphic systems, as well as to explore the benefits of the auditory processing combined with other sensory information, like vision, in robotics. To achieve these general objectives, the following set of specific objectives were proposed:

Specific objectives:

1. Open-source Neuromorphic Auditory Sensor (NAS) for FPGAs and ASICs implementations:
   (a) Study of how NAS works, its parameters, and the behavior of its internal modules when an input stimulus is presented.
   (b) Development of a new tool for automatically generating HDL files to implement a NAS according to a list of configuration parameters specified by the user.
   (c) Study of the digital ASIC design workflow from RTL description to the signoff process.
   (d) Design of an ASIC containing a tiny NAS model as a proof-of-concept of the first implementation of a digital, fully event-based neuromorphic auditory sensor in a chip.
2. Neuromorphic models for binaural cues extraction and sound source localization:
   (a) Study the biological and physiological principles of the binaural hearing in humans from the ear to the auditory cortex, and the effect of the visual attention to the auditory perception.
   (b) Study and analysis of state-of-the-art Jeffress model implementations for the ITD extraction.
   (c) Design and development of a digital, event-based Jeffress model for FPGAs.
   (d) Design and development of a digital, event-based model of a simplified human Superior Olivary Complex (SOC) model for performing the binaural cues extraction.
   (e) Integration of the proposed SOC model with the NAS model in order to contribute to the implementation of a neuromorphic hearing sense.
   (f) Characterization and evaluation of the integration under several conditions for analyzing its scalability and adaptability.
   (g) Study of novel alternatives of binaural cues extraction by taking inspiration from the optical flow estimation process in insects.
   (h) Design and implementation of the TDE neuron model for FPGAs.

   (i) Test, analysis and evaluation of the use of the TDE model for the ITD extraction as an alternative to the Jeffress model for the sound source lateralization task.

3. Integration of neuromorphic auditory systems for real-time audio processing in robotics:
   (a) Study of the relation between the sound recognition task and Central Pattern Generators (CPGs) in biology and the state-of-the-art implementations.
   (b) Design of an audio-guided SCPG based on the central nervous system of insects on SpiNNaker.
   (c) Implementation of a neuromorphic robotic platform for performing real-time test about locomotion changes depending on the recognition of different auditory input stimuli.
   (d) Study of the audio-visual integration process in the brain.
   (e) Design, implementation and test of a neuromorphic robotic system to integrate event-based sensory information for attention-related tasks.
   (f) Integration of the proposed neuromorphic auditory sensor within the iCub robotic platform.
   (g) Evaluation of the integration, identification of improvements, and implementation of a demonstration of real-time audio processing in iCub.

2.2 Thesis structure

The thesis is structured in three chapters, following the specific objectives previously detailed, as well as a conclusions and future works chapter. In addition, three appendices are included containing useful information for the reader, such as software manuals and design files. Each chapter and appendix are briefly described below:

- Chapter 3: Open-source Neuromorphic Auditory Sensor (NAS) for FPGAs and ASICs implementations.

  In this chapter, OpenNAS tool is presented. This tool has been developed to provide an open-source alternative of an event-based auditory sensor to neuromorphic researchers. The tool architecture, workflow, and performance are described also in this chapter. In addition, the preliminary results of the first ASIC implementation of a NAS are showed.

- Chapter 4: Neuromorphic models for binaural cues extraction and sound source localization.

This chapter can be split in two parts. The first part is focused on the implementation of an event-based SOC model for FPGAs. An event-based design of the Jeffress model is implemented taking into account the state-of-the-art implementations, and its characterization results are analyzed in this chapter. In the second part, a novel digital implementation of the TDE neuron is detailed. Furthermore, a proof-of-concept the ITD extraction from auditory information is provided, and its potential use as an alternative to the Jeffress model is also discussed.

- Chapter 5: Integration of neuromorphic auditory systems for real-time audio processing in robotics.

  In this chapter, three different cases of use of the NAS in robotics are presented. For each case of use, the main system components are described, as well as the input stimuli and the expected behavior. Those cases of use are: 1) an hexapod robot with audio-guided locomotion control, 2) a mobile robot with event-based sensory integration of both visual collision avoidance and sound source localization, and 3) the iCub robot with sound recognition and localization.

- Chapter 6: Conclusions and future works.

  This chapter summarizes the contribution of this work and explains the conclusions extracted from the research of this thesis. Furthermore, it lists the accepted papers and the future works in the research line of this thesis.

- Appendix A: OpenNAS tool manual.

  It contains screenshots of OpenNAS tool screens, as well as a short version of the user manual describing how the tool works and the meaning of its configurable parameters.

- Appendix B: NASIC PCB design files.

  This appendix contains both the schematic and board files of the PCB designed for testing and characterizing the NASIC.

- Appendix C: General purpose PCBs design files.

  This appendix contains both the schematic and board files of several PCBs designed during the thesis according to the needs. The development of these boards has made possible the experiments carried out in this work.

Chapter 3

Open-source Neuromorphic Auditory Sensor

"There's a silly notion that failure's not an option at NASA. Failure is an option here. If things are not failing, you are not innovating enough."

– Elon Musk

3.1 Introduction

Artificial cochleae are sensors inspired by the way that the biological inner ear works, as introduced in Chapter 1. Several models (hardware and software) can be found in the literature, with a similar architecture. In general, they all have an input stage, where the input sound stimuli are collected; a processing stage consisting of a set of band-pass filters, commonly with a cascaded topology, and finally the output stage, where the filters' outputs are obtained and preprocessed for subsequent steps.

There are software models that implement the entire auditory periphery either by means of complex SNNs (James, 2020) or by resolving complex differential equations (Zilany and Bruce, 2006). In addition, there exist other works that only implement the functionality of the cochlea by using a filter bank of gammatone filters (Tabibi et al., 2017; Liu et al., 2013).

Hardware implementations can be also divided into two groups: reconfigurable and non-reconfigurable architectures. For the latter, the implementation is fixed (analog (Yang et al., 2016) or digital silicon (Xu et al., 2018b)) and only a few parameters can be tuned, such as the filter's gain or the filter's quality factor.

Both approaches (software and hardware) present similar disadvantages when trying to use them in different situations. For instance, software models may not be deployable in other platforms due to the lack of resources or architecture incompatibility, like in (James, 2020) where the SpiNNaker machine (Furber et al., 2014a) is needed. Furthermore, hardware solutions could be expensive in some cases, or even do not match the application's requirements.

In response to this, one of the main contributions of this thesis is the introduction of the first software tool for automatically generating a full-custom Neuromorphic Auditory Sensor (NAS) for FPGA, which was presented in (Jimenez-Fernandez et al., 2017). This open-source tool is able to generate the VHSIC Hardware Description Language (VHDL) code needed for implementing a NAS. Since the generated files contain the source code of the sensor, the users can adapt, tune, and modify the model according to their needs.

Furthermore, as part of the continuous development of the NAS design flow, the first NAS ASIC, also called NASIC, was designed and fabricated as a proof-of-concept.

3.2 OpenNAS tool

One of the main advantages of developing a NAS in an FPGA is its flexibility and versatility. However, these can eventually become a serious disadvantage, as the complexity of design building and parameter tuning increases the difficulty of designing a NAS. With the aim of distributing the NAS along the neuromorphic research community, we present this open-source tool, known as OpenNAS [1] (Gutierrez-Galan et al., 2021b).

OpenNAS builds a design by instantiating its different blocks and automatically computing all the parameters (including different filter gains, cut-off frequencies, First In, First Out (FIFO) memories, and interfaces), guiding users step by step along the process.

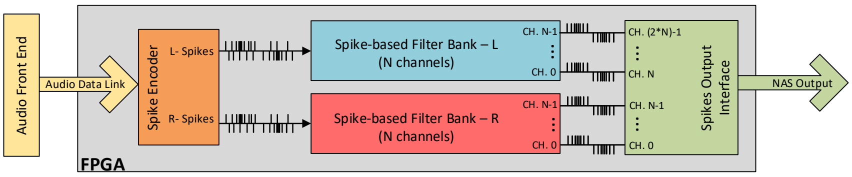


FIGURE 3.1: Block diagram of the complete architecture of a binaural NAS.

NAS is currently used for several neuromorphic applications developed by different international research groups, demonstrating the utility of OpenNAS for setting up custom projects and its integration within them. This includes pattern recognition in audio samples (Dominguez-Morales et al., 2017b) and sound source localization (Schoepe et al., 2019).

[1]https://github.com/RTC-research-group/OpenNAS

3.2.1 NAS architecture and design flow

The NAS architecture is mainly composed of three blocks, presented in Fig. 3.1. Firstly, the input audio signal is acquired by an audio front-end that converts audio information to Pulse-Frequency Modulation (PFM) spike-coded signals. Next, the spikes generated by the first block excite a Spike-based Filter Bank (SFB), which decomposes the information in different frequency bands using a cascade-fashion or parallel-fashion processing architecture (user selectable) (Jimenez-Fernandez et al., 2017).

Finally, the spikes obtained from the output of the SFB are collected by a neuromorphic output interface to propagate the NAS information to any following processing layer. The current OpenNAS version supports a parallel AER monitor as output, as well as the SpiNNaker interface, which is used to connect the NAS to a SpiNNaker board (Painkras et al., 2013).

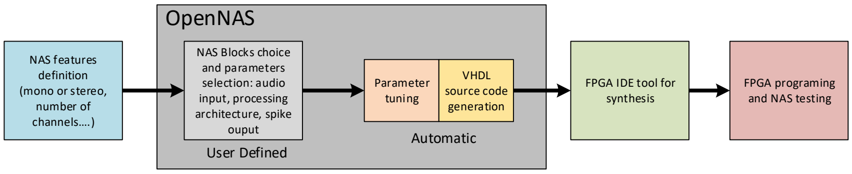

OpenNAS
NAS features definition (mono or stereo, number of channels....)
NAS Blocks choice and parameters selection: audio input, processing architecture, spike ouput
User Defined
Parameter tuning
VHDL source code generation
Automatic
FPGA IDE tool for synthesis
FPGA programing and NAS testing

FIGURE 3.2: Design flow diagram for full NAS synthesis.

Users should follow the design flow presented in Fig. 3.2, adjusting the settings of the three blocks to configure a new NAS. After this step, NAS' parameters will be computed and the source code will be generated. Finally, using a development suite for FPGAs, such as Xilinx's Vivado or Altera's Quartus, the NAS can be synthesized and deployed into an FPGA. A detailed explanation of the technical details is available in the OpenNAS wiki[2].

3.2.2 Software architecture

To represent NAS' components, we used a set of classes, where each class contains all the parameters and HDL information of a specific component. The class hierarchy is presented in Fig. 3.3. The main NAS class is "OpenNasArchitecture", which contains an instance of common parameters (OpenNASComponents) and one attribute for each of the three NAS components: *AudioInput*, *AudioProcesingArchitecture*, and *SpikesOutputInterface* (Fig. 3.3 mid).

These are abstract classes that inherit from the "HDLGenerable" abstract class (Fig. 3.3 top), which contains the methods for generating the HDL code. Finally, specific component classes inherit from *AudioInput*, *AudioProcesingArchitecture*, and *SpikesOutputInterface*, implementing each of the component features. Using

[2]https://github.com/RTC-research-group/OpenNAS/wiki

this inheritance tree, a NAS is fully modeled and structured, ready for future expansions with new NAS features.

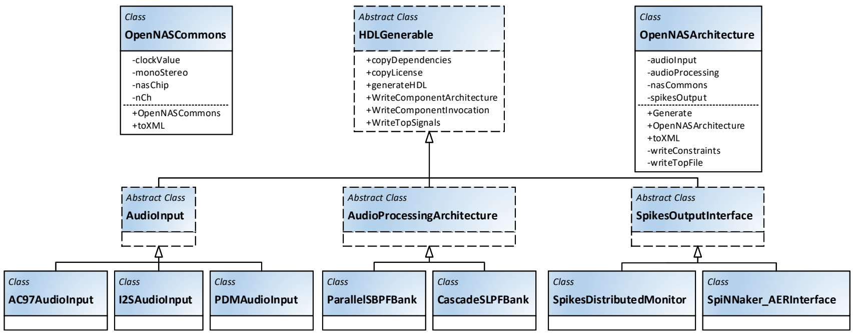


FIGURE 3.3: OpenNAS class diagram.

To guide the user, OpenNAS implements a wizard-based GUI written in Windows Presentation Foundation (WPF) (Nathan, 2006). The last step is VHDL generation, which performs the following steps:

1. Each NAS component copies its HDL dependency files and top entity to an output destination folder.
2. OpenNasArchitecture creates the top NAS HDL file.
3. Sequentially, each component writes I/O signals in the NAS top file.
4. Interface signals between components are added to the NAS top file.
5. Sequentially, each component writes its top component architecture.
6. To the NAS top file, each component adds an invocation to its instance, and these are connected to each other using interface signals.
7. A template for constraint files is generated with all NAS I/O signals.
8. A NAS summary is written as an eXtensible Markup Language (XML) file.
9. Finally, a Tool Command Language (TCL) file is generated for automatically creating the project and starting the synthesis process.

3.2.3 OpenNAS execution results

To measure the tuning error and execution time, different NASs were generated with different CPUs. The tuning error was defined as the difference between the theoretically ideal parameters' values and the final achieved values after configuring the modules' parameters. The theoretical values are obtained from the frequency parameters set in OpenNAS, where a logaritmic distribution over the frequency range is generated. The calculated values are based on the number

TABLE 3.1: OpenNAS performance comparison between two different processors.

NAS	SFB Error	AMD Ryzen 3900X (3.80 GHz)	Intel Core i7 6700HQ (2.60 GHz)
32ch. Stereo	0.48%	63.48 ms	139.14 ms
64ch. Mono	0.51%	64.36 ms	156.81 ms
128ch. Mono	0.53%	84.62 ms	247.4 ms
256ch. Mono	0.55%	127.28 ms	313.38 ms

of bits of the signals, as well as the characteristic equation of each VHDL module, which were deatiled in (Jiménez Fernández, 2010). The results are presented in Table 3.1. The generated NASs have an average error of around 0.5%, which is lower than the error reported in (Jimenez-Fernandez et al., 2010) (around 1.573% for a 64-channel NAS).

The time that the software takes to generate a NAS was measured, including internal parameter tuning, with two different processors: AMD Ryzen 3900X and Intel Core i7 6700HQ. For all the different cases, the generation time is below a few hundred milliseconds, increasing with the number of NAS' channels.

3.2.4 Conclusions

The main contribution of this work is a novel Intellectual Property (IP) core generator tool that allows researchers to easily design their own NAS for specific applications. Thanks to its friendly interface and the automatic computation of its parameters by only following 5 steps in a GUI, the NAS architecture can be freely distributed to the neuromorphic community, ready for low-cost FPGAs.

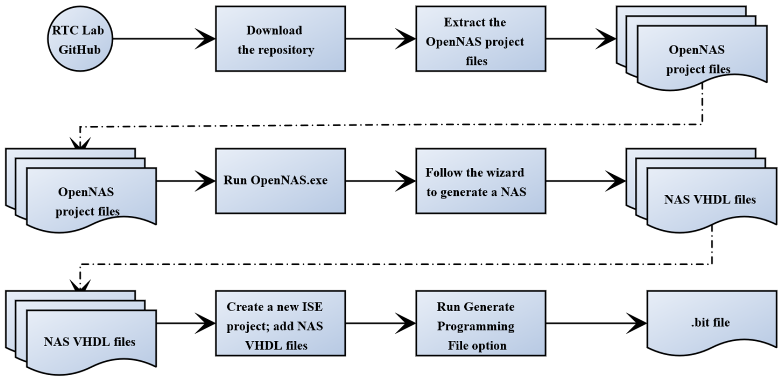

RTC Lab GitHub
Download the repository
Extract the OpenNAS project files
OpenNAS project files
OpenNAS project files
Run OpenNAS.exe
Follow the wizard to generate a NAS
NAS VHDL files
NAS VHDL files
Create a new ISE project; add NAS VHDL files
Run Generate Programming File option
.bit file

FIGURE 3.4: OpenNAS tool usage flow.

OpenNAS was designed following a hierarchical class structure to represent NAS' components, with its HDL description and parameters, allowing developers to increase OpenNAS components and functionalities easily. Fig. 3.4 shows a summary of the OpenNAS usage flow from the download of the repository until the FPGA programming.

3.3 NASIC

As introduced in Section 3.2, FPGA-based solutions can be considered as one of the best options for testing and validating research developments and proof-of-concept designs. This approach is also efficient in cases that the reconfigurability of the design plays a key role, since this kind of platforms are capable of updating part of its deployed design in real time (Liu et al., 2009; Sabena et al., 2014; Montealegre et al., 2015).

Nevertheless, ASIC solutions could provide advantages for embedded applications due to its low-power consumption, small size, and application-specific input/output interfaces. For instance, a NAS ASIC would be needed in the future for the first generation of neuromorphic cochlear implants (Lande et al., 2000). In addition, ASICs are desirable in robotic platforms, like iCub (Natale et al., 2017), where many different parts and components have to fit in a reduced space.

For this purpose, this thesis presents an ASIC which implements a tiny model of the Neuromorphic Auditory Sensor (NAS) as a proof-of-concept, also known as NASIC. The NAS was generated using the OpenNAS tool, and to the best of our knowledge, it is the first time that a full event-based, digital Neuromorphic Auditory Sensor (NAS) is produced as an ASIC. The project's repository is open-source, and it can be found on GitHub [3].

3.3.1 Motivation

The NAS was already used within robotic applications, either integrated or not into the platform itself, as presented in (Schoepe et al., 2019; Gutierrez-Galan et al., 2019a; Cerezuela-Escudero et al., 2018). In those cases, neither the power consumption nor the high performance was a crucial aspect to take into account since the main goal was the applications themselves.

Recent works in the field of e-health and biomedical engineering, where the NAS was the main component (Dominguez-Morales et al., 2017b; Dominguez-Morales et al., 2021a), motivated the decision of designing, fabricating, and testing a NAS ASIC. This way, it would be possible to think of commercial applications including neuromorphic chips for health-related tasks in hospitals.

[3]`https://github.com/dgutierrezATC/NASIC`

In this sense, analog neuromorphic designs are somehow forced to be produced as ASICs to be tested and used in real-time, low-power applications. Although there is reconfigurable alternatives, as the Field-Programmable Analog Array (FPAA) (George et al., 2016) that offers the possibility to deploy custom analog designs in a reconfigurable floating-gates fabric, researchers often prefer to fabricate their own chips to really take advantage of the technology, such as the low-power consumption (Milde et al., 2018). For instance, there are analog cochleae ASICs, as presented in Section 1.3.2.1.3, which have been successfully used in many projects, demonstrating their low-power features and good performance.

Therefore, we would like to compare the performance of our proposed digital, fully event-based cochlea model in an ASIC with the existing state-of-the-art silicon cochleae.

Finally, one of the main goals is to study the ASIC design flow for adding new functionalities to OpenNAS in order to automatically generate common ASIC design files, such as the constraints file or testbenches. This way, the OpenNAS tool would be involved in future NASIC design and fabrication.

3.3.2 Base architecture

The OpenNAS tool was used for generating the NAS' HDL files. Since this was the first ASIC designed by the RTC Research Lab., a tiny version of a NAS was desired to minimize potential risks, such as timing problems, size restrictions, etc., and to reduce simulation and synthesis times. As it was mentioned before, the main goal is to check whether the performance of an ASIC-based NAS is similar or not when compared to the performance of an FPGA-based NAS. Table 3.2 summarizes the selected values for the base project, organized according to the OpenNAS parameters.

An eight-frequency-channels, mono, cascade Neuromorphic Auditory Sensor (NAS) was generated, setting the working frequency range between 20 Hz and 22000 Hz. The main clock frequency was set to 48 MHz, that is the same clock frequency of the FPGA-based board ZTEX 2.13, which has been used to test the same files on FPGA prior to design the chip. As input interfaces, both I2S-based interface and Pulse-Density Modulation (PDM)-based microphones were added for testing the NAS' behavior with different stimuli sources.

As output interface, a distributed events monitor was included for providing an AER interface to send the NAS' output events. However, this module was removed from the design due to timing problems to satisfy the timing constraints. Therefore, no output interface was included in this first version. Instead, two dedicated pins are used for each filter's output, having sixteen pins in total for the output events (eight frequency channels, with positive and negative events for each channel). Thus, the absence of events monitor implies the use of an

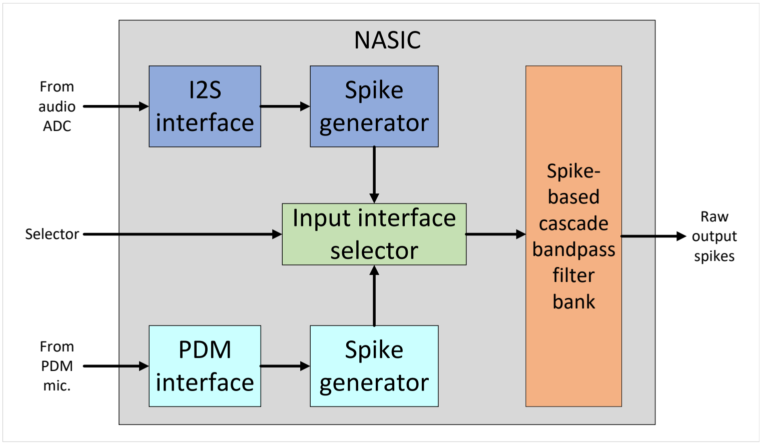


FIGURE 3.5: NASIC's internal architecture block diagram.

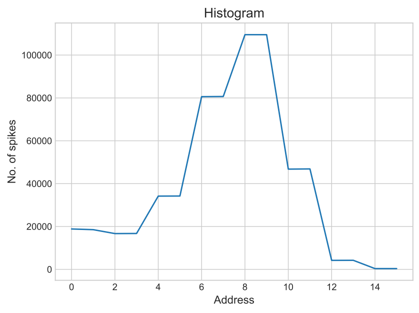
external FPGA-based board for implementing both the events monitor and the AER protocol.

TABLE 3.2: NASIC features

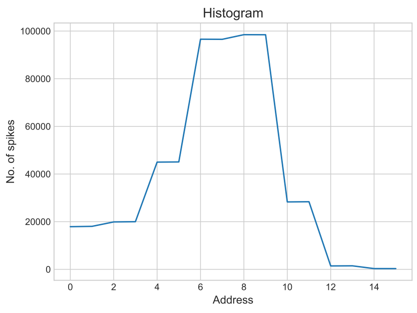
	Parameter	Value	Notes
NAS common settings	NAS chip	Other	180 nm TSMC
	NAS type	Mono	
	Num. of channels	8	
	Clock freq. (MHz)	48	
Input interface	Audio input	I2S + PDM	All the values were by default.
Processing architecture	NAS architecture	Cascade	
	Start frequency	20 Hz	
	End frequency	22000 Hz	
	SLPF output att.	-15 dB	
Output interface	Spikes output	Raw	No AER monitor was integrated.

The NAS base project was deployed into an FPGA and characterized in order to set a ground truth for being used as reference when validating the ASIC. For this characterization, a set of six pure tones were generated and used as input of the NAS. The pure tone frequencies are identical to the ones used in (Dominguez-Morales et al., 2016).

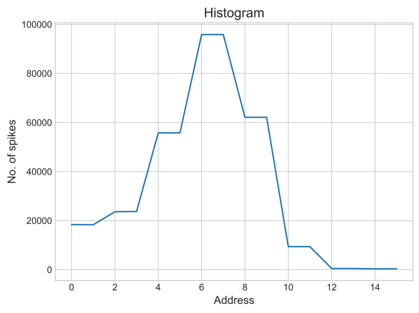
The duration of each pure tone was two seconds, and the computer's volume was set to 60%. A high speed audio Analog-to-Digital Converter (ADC) with Integrated Interchip Sound (I2S) interface was used, since it allows to connect an audio cable directly from the computer to the NAS. Fig. 3.6 shows the histograms of the pure tones used.



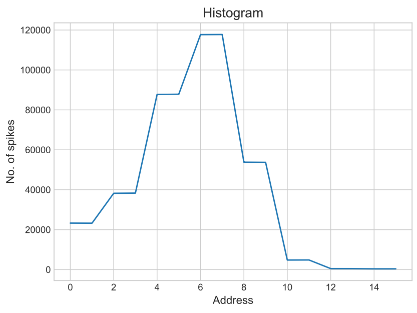
(A) 261 Hz.



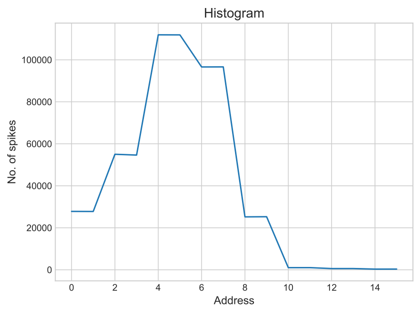
(B) 349 Hz.



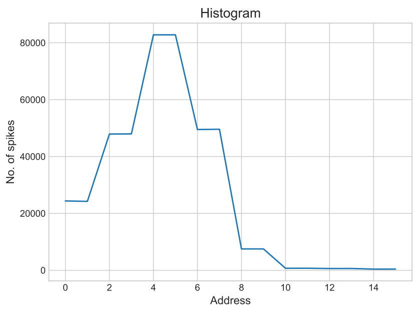
(C) 523 Hz.



(D) 698 Hz.



(E) 1046 Hz.



(F) 1396 Hz.

FIGURE 3.6: Histograms from pure tones using NAS on FPGA. Middle frequencies associated to each frequency channel were: channel 0, 22000 Hz; channel 1, 8090 Hz; channel 2, 2975 Hz; channel 3, 1094 Hz; channel 4, 402 Hz; channel 5, 148 Hz; channel 6, 54 Hz; and channel 7, 20 Hz.

As it can be seen, the main activity peak for each case is shifted from low frequencies (higher addresses) to high frequencies (low addresses) according to the input stimulus frequency. The shape of the histogram keeps almost identical in all cases. However, the average activity, expressed as the number of events per second, varies between cases. The case 3.6a had about 325000 events per second, while the case 3.6f had around 250000 events per second.

This effect is produced due to, for low frequency sounds, the lasts bandpass filters of the cascade filter bank are the ones tuned to have a better response to those sounds, in the same way to the basilar membrane. Therefore, the sound will go through the filters in the filter bank, exciting all of them and, thus, producing more spikes. In the opposite way, high frequency sounds will stimulate the filters placed at the beginning of the filter bank, thus not stimulating the low-frequency tuned bandpass filters, which are placed at the end of the filter bank.

Fig. 3.7 shows the sonogram of each pure tone obtained from the NAS model before designing the ASIC. The correlation between the histograms and the sonograms is perceptible. It can also be seen how the main frequency band, i.e., the frequency band with the higher activity, changes according to the input sound, as it was already observed in the histograms.

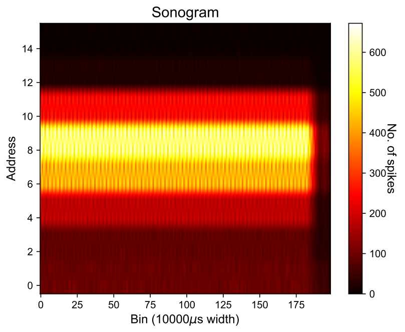
3.3.3 NASIC design process

In this design, a 180 nm process technology from Taiwan Semiconductor Manufacturing Company (TSMC) was used. As Electronic Design Automation (EDA) software, Design Compiler from Synopsys was used for performing the synthesis, while Innovus from Cadence was used for carrying out the physical implementation. The simulations were run using ModelSim from Intel FPGA.

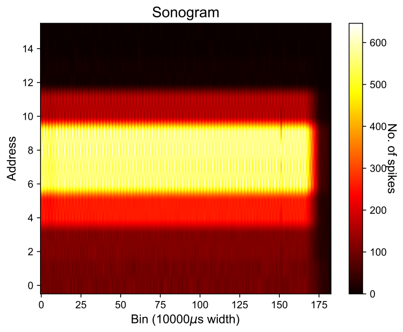
First, the synthesis tool, Design Compiler in this case, was set up. The search paths for specifying the directories which contain the core and I/O libraries were set, among others. After configuring the synthesis tool, the next step was to load the design. This process is a two-stage process. The first one is the analysis stage, where the software analyzes the RTL HDL code and compiles it. If the design had any problem, an error would be raised.

The second stage is the design elaboration, where the tool, after analyzing the code, creates the design and loads it in memory. In this process, much information and feedback is shown, including state machine extraction, register inference, and design hierarchy. Several changes were carried out due to some design errors found at the elaboration step. At this point, a netlist was generated but without any link to the process technology library.

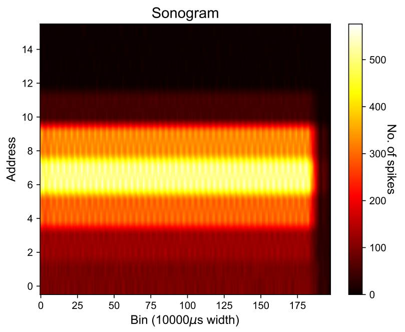
As a first check, this generated netlist was used as the main component in the simulation. First, the original HDL code generated by OpenNAS was simulated, and the resultant simulation file, which includes the NAS' event address and its timestamp, was saved. Then, the elaborated netlist was also simulated. Finally, both simulation files were compared, perfectly matching in both addresses and timestamps values.



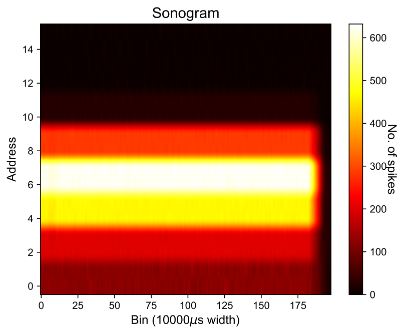
(A) 261 Hz.



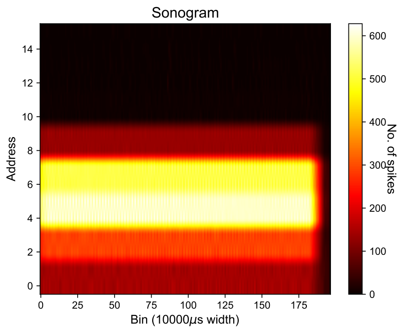
(B) 349 Hz.



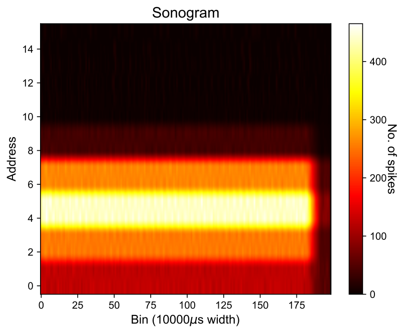
(C) 523 Hz.



(D) 698 Hz.



(E) 1046 Hz.



(F) 1396 Hz.

FIGURE 3.7: Sonograms from pure tones using NAS on FPGA.

When the generic netlist was ready, the mapping (to cells) and optimisation procedures could be applied. Since the timing constraints were not specified at this point, no timing optimisations were performed. Instead, an area reduction optimisation was done, having a first approximation of $0.5 \times 0.5mm^2$.

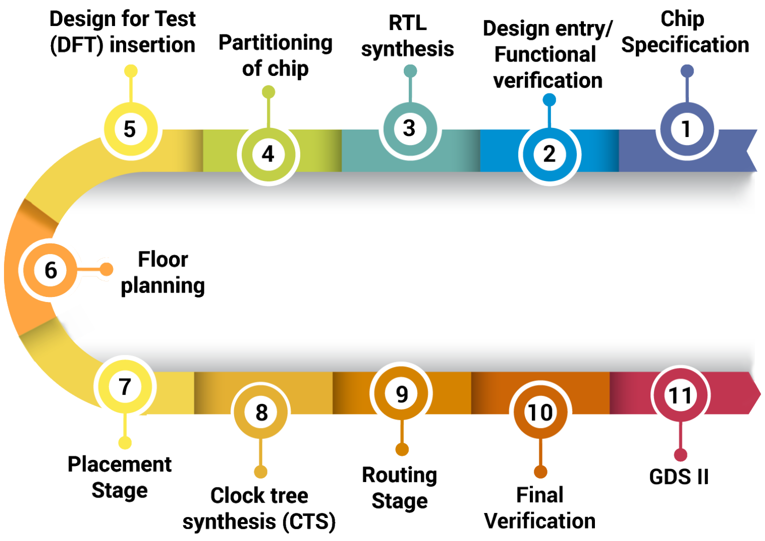


FIGURE 3.8: ASIC design workflow. Image taken from https://www.einfochips.com/

The next step was to constraint the design by modeling the clock and constraining both the inputs and outputs. First, the clock was created, being defined at 48 MHz. Just after the clock definition, a first timing test was performed, verifying that most of the paths were constrained by the clock. Then, the rest of the clock effects were modelled by defining the clock latency, the clock uncertainty, and the clock transition. Finally, the I/O timing constraints were set, and a timing verification was run.

Several timing problems were identified at this point, which were located in two different parts of the design. On the one hand, there was inconsistency with the reset signal, which was used sometimes low active and sometimes high active. On the other hand, the AER events monitor, that was originally integrated on the design, presented timing problems since its FIFO memories were implemented as registers instead of memory cells, as it is synthesized in an FPGA. The reset issue was solved, but the events monitor problem was not solved and, therefore, it was removed.

After constraining the design, and once the timing problems were solved, a timing optimisation was performed. A new simulation was carried out in order to check that the changes applied to the design did not affect its behavior, being the result of the simulation identical to the reference.

The procedures and function calls were automated by using TCL scripts, allowing us to repeat the whole design flow every time a change was done. These TCL scripts are being integrated in OpenNAS with the aim to facilitate the NASIC design process in the future.

For performing the floorplan step, we moved from Design Compiler (Synopsys) to Innovus (Cadence). Therefore, the design was exported from the first software tool and imported in the second software tool. The global configuration was created, as well as the physical constraints (core utilization, boundaries, power ring, etc).

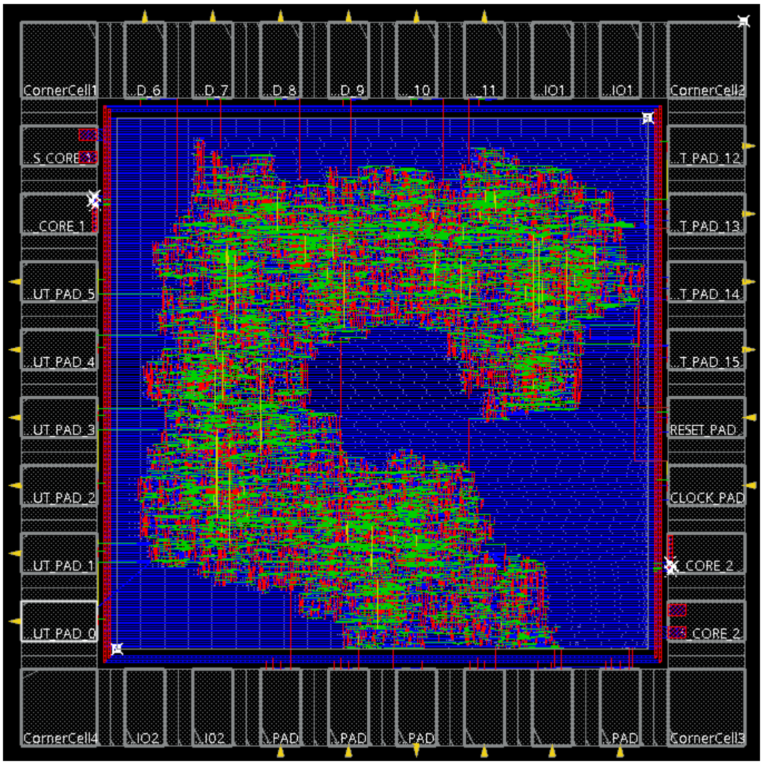


FIGURE 3.9: NASIC cells distribution after the place & route process.

Input and output pads (known as IO pads) were also defined and placed at this stage. In turn, IO pads could be split depending on their use, i.e., if the pad was going to be used for the clock, power supply, or for other signals. Fig. 3.9 shows the distribution of the IO pads on the chip.

After the physical constraints were set, the place & route process was carried out. The design's cells were loaded, placed, and distributed over the core area available, and routed. Although the area estimation during the synthesis was $0.5 \times 0.5mm^2$, a core size of $1 \times 1mm^2$ was set to facilitate the place & route process. Finally, the cells were connected to either the IO pads or power supply rings, if needed, distributed over the core as shown in Fig. 3.9.

For verification purposes, another simulation was performed after the place and route process. In this case, the cells' library was used, containing the real timing information of the physical cells. Both the number of output spikes and their order were identical compared to the reference. However, small time differences (a few nanoseconds) were obtained.

The last step was to fill the empty space inside the core with filler cells to avoid planarity problems and to ensure good continuity. After that, the final compilation was performed applying area, power, and timing optimizations.

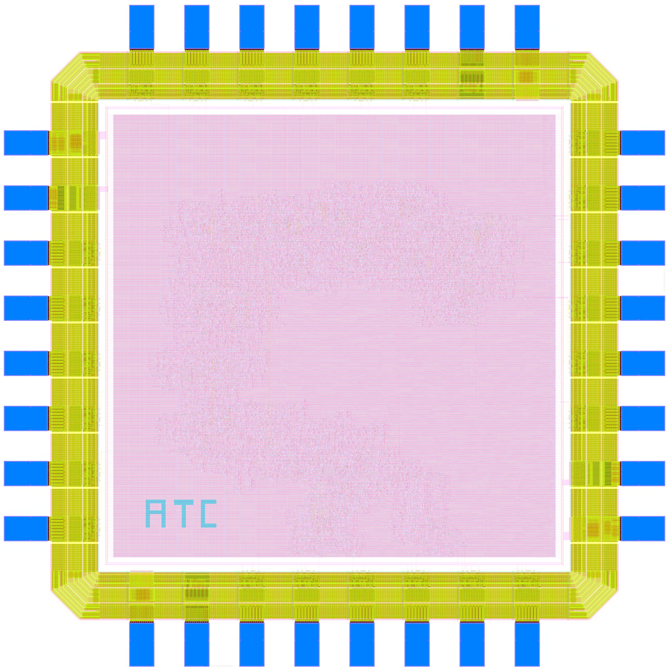


FIGURE 3.10: NASIC layout.

Fig. 3.10 shows the final design layout, including the bonding pads. Thirty two pins were needed according to the number of IO pads used during the ASIC design process. However, a sixty four pins Ceramic Quad Flat Package (CQFP) package was used due to packaging limitations.

Table 3.3 summarizes the NASIC's pinout. As it can be observed, half of the pins were not connected internally to any circuit. For simplification purposes, the pin usage distribution was set in an alternate way, leaving one empty pin between each used pin.

TABLE 3.3: NASIC chip pinout.

Pin number	Pin name	Type	Typical value	Notes
0	VSS_CORE_2	Input	GND	Digital GND.
2	VDD_CORE_2	Input	1.8V	Input current up to 1 A.
4	SPIKES_OUT_5	Output	3.3V	
6	SPIKES_OUT_4	Output	3.3V	
8	SPIKES_OUT_3	Output	3.3V	
10	SPIKES_OUT_2	Output	3.3V	
12	SPIKES_OUT_1	Output	3.3V	
14	SPIKES_OUT_0	Output	3.3V	
16	VSS_1	Input	GND	Digital GND.
18	VDD_1	Input	3.3V	Input current up to 1 A.
20	PDM_I2S_SEL	Input	3.3V	0 for I2S; 1 for PDM.
22	PDM_DATA_LEFT	Input	3.3V	
24	PDM_CLK_LEFT	Output	3.3V	
26	I2S_SD	Input	3.3V	
28	I2S_WS	Input	3.3V	
30	I2S_SCLK	Input	3.3V	To add a 2.7 kOhm pull-down resistor.
32	VSS_CORE_1	Input	GND	Digital GND.
34	VDD_CORE_1	Input	1.8V	Input current up to 1 A.
36	CLK	Input	3.3V	The clock frequency is 48 MHz.
38	RST	Input	3.3V	Low active reset.
40	SPIKES_OUT_15	Output	3.3V	
42	SPIKES_OUT_14	Output	3.3V	
44	SPIKES_OUT_13	Output	3.3V	
46	SPIKES_OUT_12	Output	3.3V	
48	VSS_1	Input	GND	Digital GND.
50	VDD_1	Input	3.3V	Input current up to 1 A.
52	SPIKES_OUT_11	Output	3.3V	
54	SPIKES_OUT_10	Output	3.3V	
56	SPIKES_OUT_9	Output	3.3V	
58	SPIKES_OUT_8	Output	3.3V	
60	SPIKES_OUT_7	Output	3.3V	
62	SPIKES_OUT_6	Output	3.3V	

3.3.4 ASIC validation and characterization

After finishing the design part, the fabrication files were sent to the foundry. From all the manufactured ASICs, ten of them were packaged also by the foundry. No functional test were performed before receiving the ASICs, so a custom PCB was needed in order to check and validate each ASIC functionality and to characterize its behavior.

Since the core cells and IO cells used in this design needed different operating voltages (1.8 V and 3.3 V, respectively), two linear regulators were needed in the NASIC test PCB for converting the 5 V input to both voltages. This voltage could be obtained from two different sources: 1) from an external power supply or 2) from the FPGA to which the test PCB is connected. In the same way, a 48 MHz clock was needed, which could be taken from an external oscillator or from the FPGA.

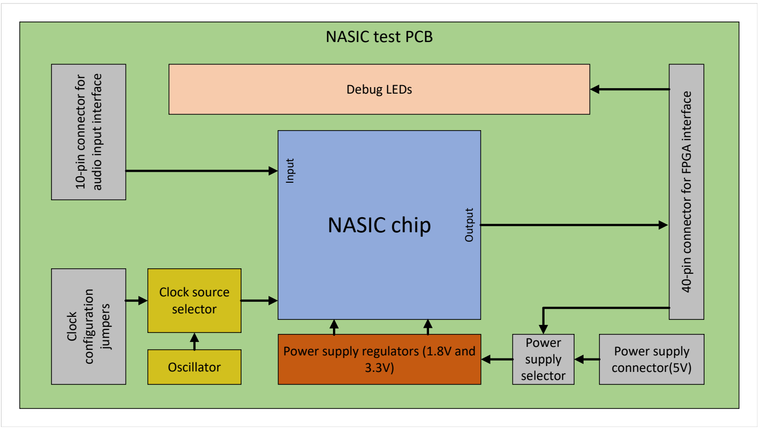

NASIC test PCB
10-pin connector for audio input interface
Debug LEDs
Input
NASIC chip
Output
40-pin connector for FPGA interface
Clock configuration jumpers
Clock source selector
Oscillator
Power supply regulators (1.8V and 3.3V)
Power supply selector
Power supply connector(5V)

FIGURE 3.11: NASIC test PCB block diagram.

Related to the connectors, two of them were needed. One of them was used to interface the audio input PCB (both the schematic and the board can be found in Appendix C.1). The other one was used to interface the NASIC test PCB with the FPGA. Fig. 3.12 shows a real picture of the NASIC test PCB with the audio input board, and its schematic can be found in Appendix B.

For characterizing the chips, the same set of six pure tones were used as input stimuli for each ASIC. Audio files were played by using the same computer and the same volume in order to fairly compare the results. Fig. 3.13 shows the histograms of two pure tones (261 Hz and 1396 Hz) from three different chips.

Using the first case as example, it can be seen that both histograms 3.13a and 3.13b are identical even though the input stimuli have different frequencies. Compared to their corresponding histograms obtained from the FPGA (Fig. 3.6a and Fig. 3.6f, relatively), it can be observed that the ASIC's response does not

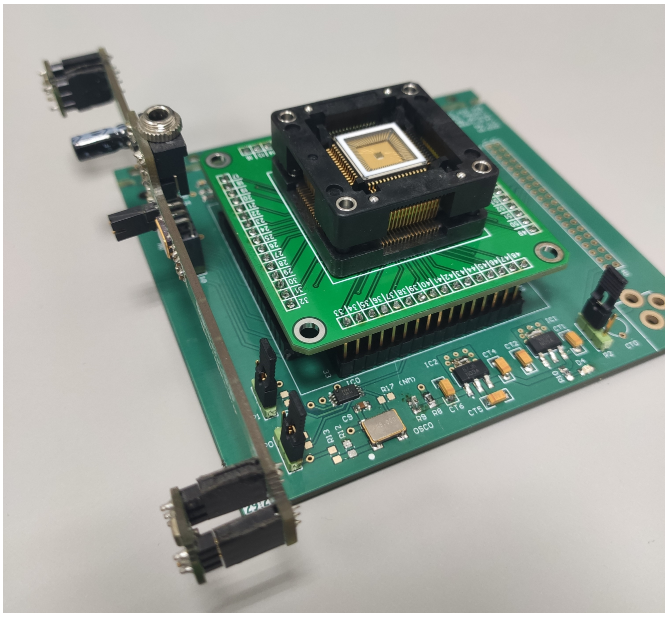


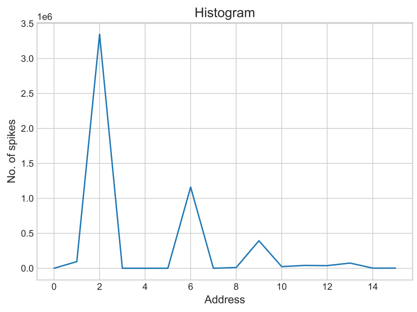
FIGURE 3.12: NASIC test PCB with the audio input board.

match with the expected output. In addition, it is important to mention that only five over ten chips produced events when an input sound was provided. Chips 1, 2, and 3 had very similar responses, while chip 4 and chip 5 had opposed responses.

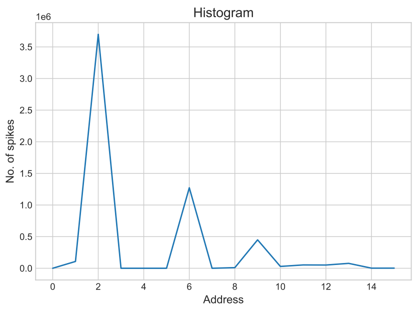
A common aspect of all the histograms is that there are channels that do not produce any event. Nevertheless, the channels that do not produce events in one specific chip often produce events in a different chip. Therefore, it can be said that this effect is due to the test PCB. In addition, it can be asserted that it is not a design problem coming from the filter bank since, in that case, if one of the first filters has a failure (e.g., not producing events) the following filters will not produce any events neither (because of the cascade architecture of the filter bank). However, Fig. 3.13c shows that the last filter produced a response to the input stimulus while the first filter did not.

The average activity for each chip was measured, obtaining about 30000, 7000, 18000, 3000, and 150 events per second for chips 1 to 5, respectively. Those distinct values indicate that there is not a fixed pattern in the chips' response, making it difficult to determine the source of the problem. Similarly to the histograms, the average activity did not vary between pure tones.

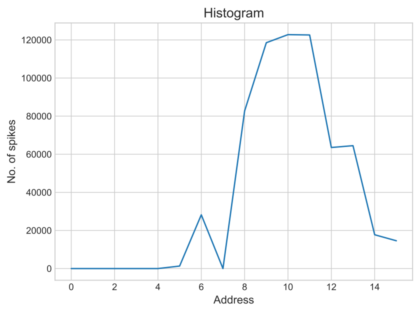
Along with the histograms, the sonograms were also generated, and they are shown in Fig. 3.14. It can be observed that the sonogram shown in Fig. 3.14f



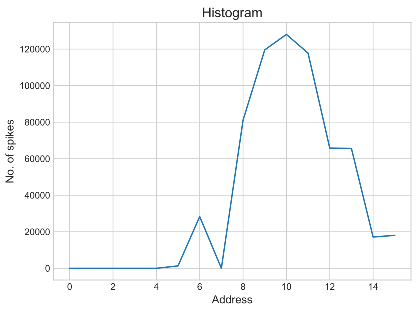
(A) Chip 1 - 261 Hz.



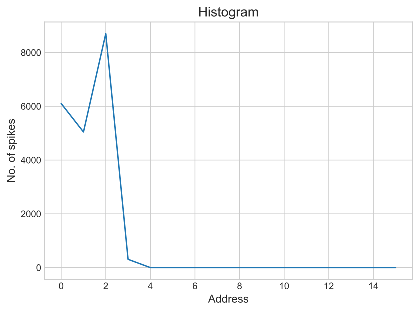
(B) Chip 1 - 1396 Hz.



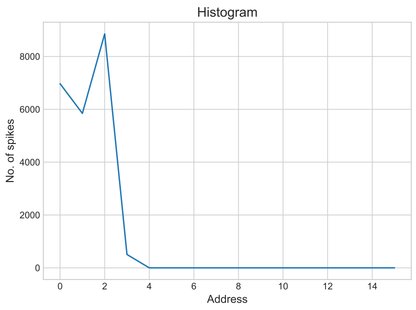
(C) Chip 4 - 261 Hz.



(D) Chip 4 - 1396 Hz.



(E) Chip 5 - 261 Hz.

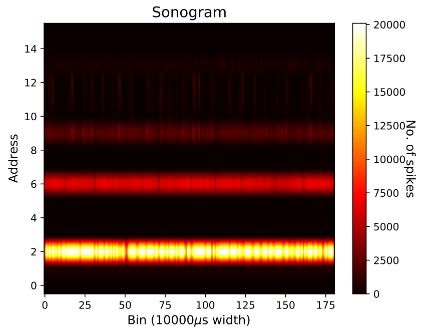


(F) Chip 5 - 1396 Hz.

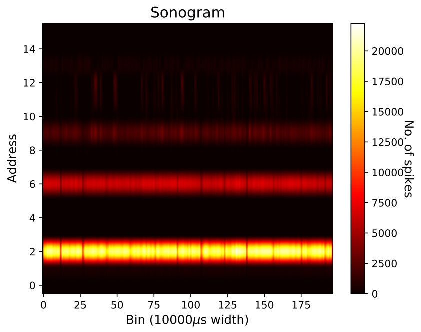
FIGURE 3.13: Histograms from pure tones using NASIC.

presents a discontinuous response to a continuous stimulus over time. Similar effect is observable in Fig. 3.14c. This problem could be related to timing aspects, being this feature critical for event-based systems.

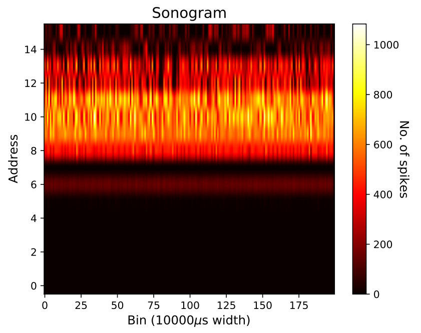
The last experiment carried out for testing the NASIC consisted in measuring the NAS response to a chirp signal. A sound signal of 10 seconds was generated, where the starting frequency was set to 100 Hz and the ending frequency was



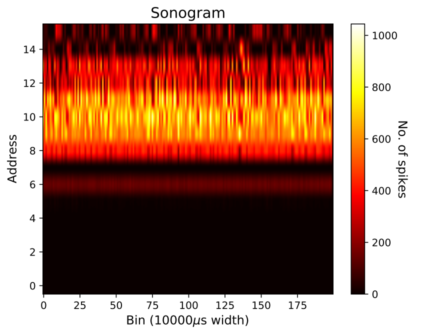
(A) Chip 1 - 261 Hz.



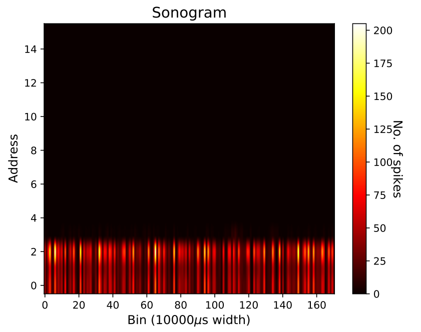
(B) Chip 1 - 1396 Hz.



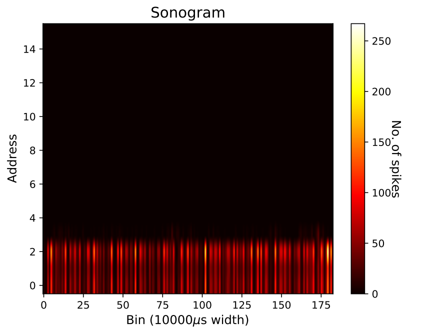
(C) Chip 4 - 261 Hz.



(D) Chip 4 - 1396 Hz.



(E) Chip 5 - 261 Hz.



(F) Chip 5 - 1396 Hz.

FIGURE 3.14: Sonograms from pure tones using NASIC.

set to 20000 Hz. This audio stimulus was presented to the FPGA-based NAS (the reference) and also to the five NASIC chips that were working.

Fig. 3.15 top shows both the sonogram and the histogram for the FPGA-based NAS results, and Fig. 3.15 bottom shows both the sonogram and the histogram for the ASIC-based NAS results (chip 3). Between all of them, only chip 3 presented a similar response in the average activity when compared to its digital counterpart, as it can be observed from Fig. 3.15b and Fig. 3.15d. In fact, although the curve seems similar, the number of events is almost one order of magnitude lower in the FPGA-based case.

Despite that similarity, the sonograms show the real response for both cases. The sonogram from the FPGA, shown in Fig. 3.15a, fits with the expected behavior of the NAS when a sound signal starts with a low frequency and starts increasing its frequency. It can be clearly observed how the activity is shifted from low frequency channels to high frequency channels. However, the sonogram from the ASIC, shown in Fig. 3.15c, did not present that behavior.

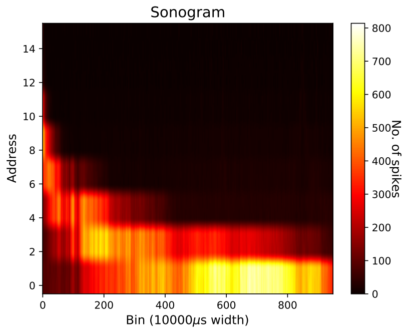
Surprisingly, the increasing and decreasing behavior of the overall activity was observable for the ASIC-based NAS while the chirp was increasing its frequency. The behavior of channel 3 was similar in both cases (FPGA and ASIC), having the main activity around time bin 200 and starting to decrease up to the end of the recording.

With these results, no further characterizations nor measurements were carried out. It was considered the power consumption measurement as not relevant since the output activity did not correspond to the input stimuli, thus not having a correlation between the power consumption and the NAS' response. Nevertheless, this first attempt to create an ASIC that implements a neuromorphic sensor helped us to both improve the NAS model and to learn more about digital design concepts.

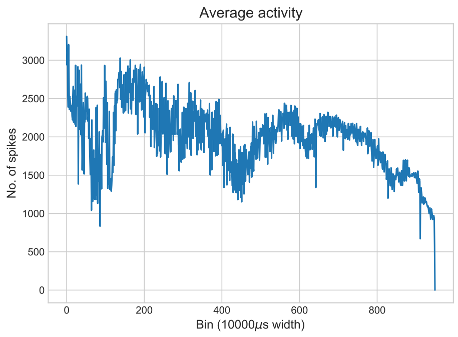
In addition, it is important to mention that the behaviors observed from the chips were not produced due to problems of the designed PCB since, in that case, a similar response would be obtained from all the ASICs. Therefore, an mistake was made during either the design flow or the assembly process where the chip was encapsulated. Nonetheless, further research about the problem will be carried out with the intention of designing the second version of the NASIC.

3.4 FPGA vs. ASIC

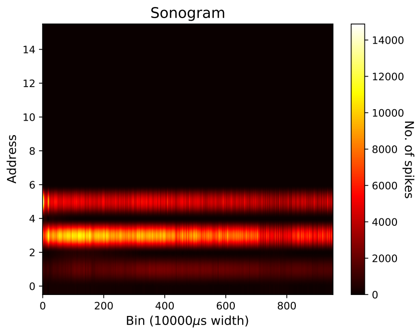
There is not a good answer to the question "What is better: an FPGA-based or an ASIC-based solution?". Like most of the questions in the field of engineering, the best answer is always "It depends on the target application". For research, to use an FPGA could be enough when a proof-of-concept application is being tested or even a real application is being characterized. However, for commercial purposes, maybe an ASIC is desired.



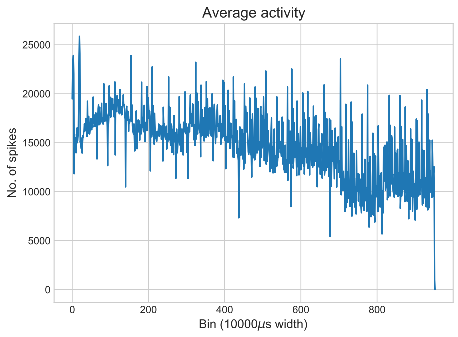
(A) Sonogram from FPGA.



(B) Average activity from FPGA.



(C) Sonogram from chip 3.



(D) Average activity from chip 3.

FIGURE 3.15: Comparison between FPGA and chip 3.

First thing to take into account is the effort (in terms of both money and time) employed to design, test, and validate an ASIC. In addition, the complexity of the process implies the risk to make mistakes. However, the control over the whole design is guaranteed even at the gate level. On the other hand, when using FPGAs, researchers can focus all their efforts on the model itself without taking into consideration the technological aspect of the hardware, saving time and, thus, money.

Therefore, the next question could be: "Is it then worth to spend time on ASIC design in research?". According to my experience, it would be better to use an FPGA-based solution while it is possible. Nowadays, FPGA chips integrates both programmable logic and microcontrollers that are able to run operating systems with low power consumption, thus allowing to have the best from both worlds in a single platform. However, an ASIC-based solution will always consume less power and will fit better with the final application, thus being the best solution when an absolute control of the system is required.

Finally, an ASIC design would be a good option for commercial purposes and applications, since mass fabrication would decrease production costs. In this case, many factors should be taken into account, like the supply and demand ratio, target consumer, sales volume, etc. Nevertheless, the fact of creating an ASIC, although it was not perfectly working or the sales volume did not reach the expectations, could open new paths for future neuromorphic solutions to audio-related problems in humans.

Chapter 4

Event-based models for the sound source localization task

"The art of conversation is the art of hearing as well as of being heard"
– William Hazlitt

4.1 Introduction

Besides the cochlea, the auditory ascending pathway contains other nuclei that extract important features from the input sound. Among them, nuclei located at the Superior Olivary Complex (SOC) are capable to measure both the Interaural Time Difference (ITD) and the Interaural Level Difference (ILD). Those measurements, also known as binaural cues, play a crucial role in the sound source localization task since they provide useful information about the auditory scene (Kandel et al., 2000).

In addition, sound source localization can be considered one of the most remarkable survival skills, since it allows to identify incoming risks in short time periods, i.e., hundreds of microseconds. In fact, auditory attention and audiovisual stimuli integration also use this capability, thus, being considered as an important part of the brain.

According to biology, the SOC in humans, shown in Fig. 4.1 is composed by three main nuclei, which are the Medial Superior Olive (MSO), the Medial Nucleus of the Trapezoid Body (MNTB), and the Lateral Superior Olive (LSO), among others. As it was mentioned in Chapter 1, the MSO extracts ITDs, while the LSO extracts ILDs. Many works can be found in the literature, either in-deep studying how those nuclei work or implementing their functionality for specific applications (Bhadkamkar and Fowler, 1993; Lazzaro and Mead, 1989b; Liu et al., 2010a; Dávila-Chacón et al., 2018).

The model proposed by Jeffress in (Jeffress, 1948) was a first approach for modelling the MSO. However, the model has been improved over the years,

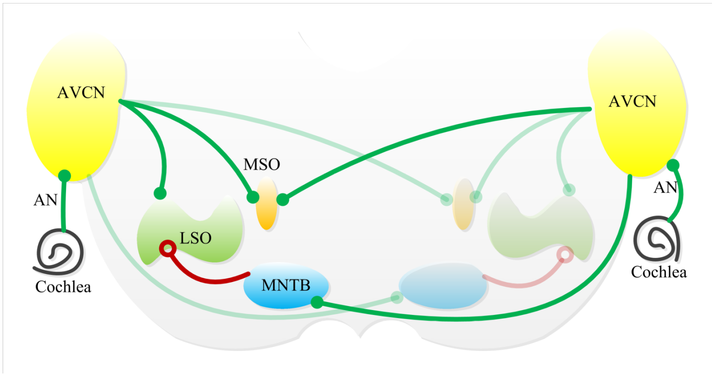


FIGURE 4.1: Superior Olivary Complex in biology. Green lines represents excitatory connections, while red lines indicates inhibitory connections. Image taken from (Liu et al., 2013).

becoming a complex nucleus. For example, contralateral (i.e., opposite side) projections have a different distribution compared to ipsilateral (i.e., same side) projections over the coincidence detector neurons. Moreover, delay lines can adapt themselves depending on the input stimuli.

Unlike the MSO, there is not a well-defined model for the LSO. Although it is known that neurons on this nucleus work like a frequency rate subtractor, it is thought that it may work like a frequency coincidence detector, thus having a similar architecture than the MSO.

Although there exist many implementations (Liu et al., 2008; Chou et al., 2019; Voutsas and Adamy, 2007), either software or hardware, of the SOC, the MSO, and the LSO models, neuromorphic solutions are becoming popular due to their event-based nature. Following the same design principle that was used during the development of the NAS, in this chapter, an event-based, FPGA-based model of the SOC is proposed with the aim of to continue the development of a more complete neuromorphic auditory sensor.

Furthermore, a novel digital implementation of the Time Difference Encoder (TDE) model was carried out and proposed as an alternative of the Jeffress model for the ITD extraction. In this case, instead of having an array of coincidence detector neurons, where in theory only one of them will fire an spike when a coincidence occurs, a single TDE neuron is used to code the ITD in the number of output spikes. Preliminary results showed the viability of this approach for performing the sound source localization task, although future works are needed in order to improve the accuracy.

4.2 Event-based model of the Superior Olivary Complex

The Neuromorphic Auditory Sensor (NAS) was implemented by using spike-based building blocks for audio signal processing. Those blocks consist of basic operational blocks that, once combined, are able to perform complex operations in the spike domain. For instance, the addition operation in the events domain is carried out by the Spike Hold&Fire (SHF) module (Jimenez-Fernandez et al., 2010), that imitates a simplified version of a LIF neuron with both excitatory (or first summand) and inhibitory (or second summand) inputs. When this module is syntethized for an FPGA, its resources consumption are negligible, as well as its power consumption.

Therefore, this design strategy was also followed for the implementation of the event-based digital model of the Superior Olivary Complex (SOC). As it was previously mentioned, the main goal is to achieve a full model of the ascending auditory pathway, starting with the cochlea and finishing with a spiking neural network which models the auditory cortex. Although a fully VHDL-based model is desired, the SNN would be first deployed in already existing neuromorphic computing platforms, like SpiNNaker (Furber et al., 2014a) or Loihi (Davies et al., 2018), instead of directly to an FPGA due to the SNN model complexity. In addition, such platforms already have development frameworks, thus making some important tasks easier, such as the learning procedure.

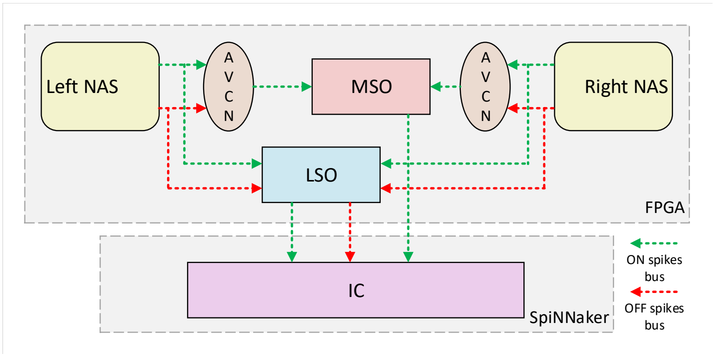


FIGURE 4.2: Block diagram of the NSSOC model for FPGA.

Taking the biological model shown in Fig. 4.1 as reference, the Neuromorphic Spike-based Superior Olivary Complex (NSSOC) model [1] has been designed and implemented as a complement for the NAS design. The main block diagram including both the NAS and the NSSOC is shown in Fig. 4.2. The NAS' output (ON and OFF events) is directly sent to two modules: 1) the

[1]The project's repository is open-source, and it can be found on GitHub https://github.com/dgutierrezATC/nssoc.

AVCN, which implements a version of the spherical bushy cells for performing the phase-lock operation, and 2) the LSO, for extracting the ILD. Then, the AVCN output phase-locked events are sent to the MSO module for extracting the ITD. All these modules were implemented by using VHDL language for further deployment into an FPGA-based board.

One of the advantages of implementing these models by using the VHDL language is that they can be directly integrated with a NAS model, thus avoiding the need to create a serial interface between the NAS and the NSSOC, e.g., an AER serial interface. Therefore, no additional delays will be added to the precise timing of the events produced by the NAS, which is important for extracting ITDs with high precision. After extracting the binaural cues, the auditory information is sent to the IC, as shown in Fig. 1.14 in Chapter 1, where all the auditory cues are combined for obtaining a spatial map of the sound sources. In the proposed model, the Inferior Colliculus (IC) was implemented as a multi-layer SNN model using PyNN and deployed into the SpiNNaker machine. The communication between the NAS, the NSSOC, and SpiNNaker is carried out through a custom PCB and a modified version of the AER-SpiNNlink Verilog module [2].

The following sections describe the implementation of each module in detail, their characterization, as well as an analysis of the performance for different MSO configurations within the NSSOC model.

4.2.1 Implementing the Cochlear nucleus

The phase lock effect takes place at the AVCN thanks to the spherical bushy cells. These cells produce an spike when the Sound Pressure Level (SPL) of the input sound is maximum, thus, having the higher displacement in the basilar membrane at that moment. Since this effect is relative to the SPL, it is not dependant on the volume of the input sound. Therefore, for a signal with the same frequency but different amplitude, the resultant phase-locked output spikes will be identical over time. Fig. 4.3 shows an example of this effect, where even though both signals increase their volume, the time difference at which the phase-locked spikes are produced are the same.

However, Kandel et al. (Kandel et al., 2000) described this effect more like a periodic response at any time of the spherical bushy cells independently of the SPL instead of an exact response at the maximum value of the SPL, as it was already shown in Fig. 1.16. Therefore, the precise timing and periodicity of the phase-locked events are considered more important than the precise moment in which they are produced.

Those two considerations were crucial when designing the phase lock module, since some simplifications were applied. As it was mentioned before,

[2]The project's repository is open-source, and it can be found on GitHub `https://github.com/dgutierrezATC/NAS_SpiNNaker_interface`.

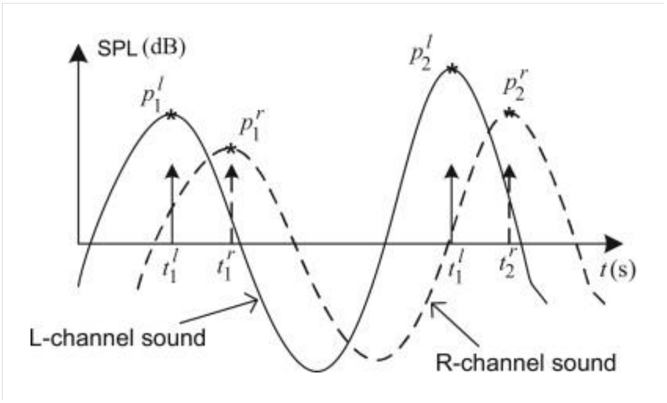
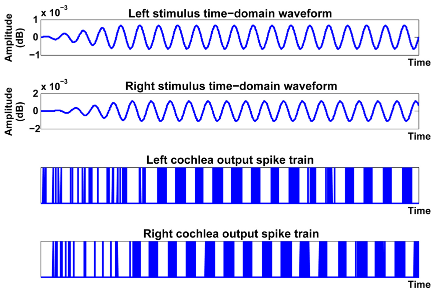
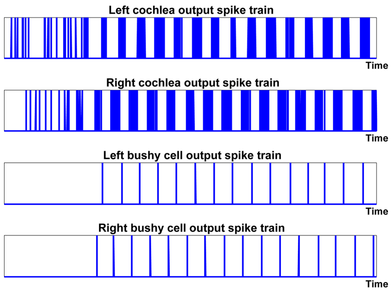


FIGURE 4.3: Representation of the phase lock effect. Image taken from (Liu et al., 2013).

the NAS' output spikes follow the rate coding, meaning that the higher the value is, the higher the frequency will be (and the lower the ISI will be). Thus, in order to detect the maximum value of the SPL, it should be enough with just detecting the minimum ISI value. Nevertheless, by applying the simplifications, detecting the zero-crossing points of the signal is enough. When using the NAS, this zero-crossing detection process can be made by only detecting the change from ON events (or positive) to OFF events (or negative), since it has both types of outputs. In other models where the output events do not have polarity, as the example shown in Fig. 4.4, the best solution would be the minimum ISI value.



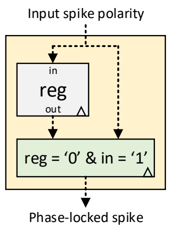
(A) Plot showing sample output data from the cochlea model.



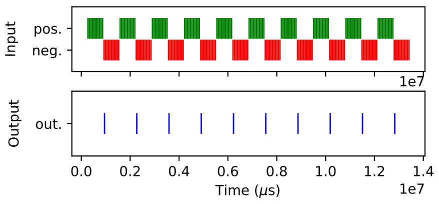
(B) Plot showing sample output data from the bushy cell neuron.

FIGURE 4.4: Example of the phase lock effect.

From the digital design point of view, the zero-crossing approach requires less hardware resources compared to the ISI analysis, and the design complexity is also reduced. Therefore, the zero-crossing model was implemented for modelling the spherical bushy cell. The model consists of a register that stores the polarity of the last produced event and a comparator that detects the polarity change from positive to negative. The output of the model is a single output that corresponds to the phase-locked event without any polarity. Fig. 4.5a shows the block diagram of the proposed model, and Table 4.1 summarizes its FPGA resources consumption.



(A) Block diagram.



(B) Simulation result.

FIGURE 4.5: Block diagram and simulation results of the spherical bushy cell model.

Firstly, the design was simulated in order to verify its functionality. A basic test, in which a sinusoidal spiking signal was artificially generated and used as input, was carried out. In consonance with the idea, the observed behavior should be to obtain an output event as soon as the first negative event is received at the input. As it can be seen in Fig. 4.5b, the simulation results match the expected behavior, as well as the two considerations which were mentioned before. In addition, the frequency of the input signal can be estimated by measuring the ISI value between two consecutive phase-locked output events, thus facilitating the sound recognition task or pure tones classification (Dominguez-Morales et al., 2016).

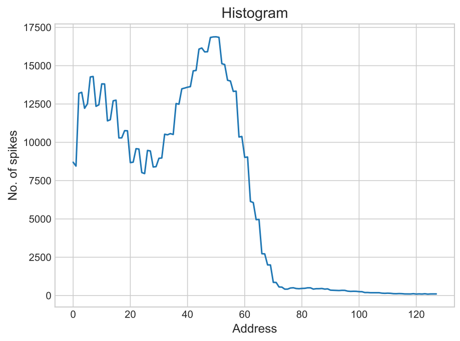
TABLE 4.1: Hardware resources utilization of the spherical bushy cell model for different FPGA devices.

FPGA chip	Slice Registers Used / Available	Slice LUTs Used / Available
XC6SLX150T (Spartan-6)	2/184304 (<0.01%)	0/92152 (0.0%)
XC7A75T (Artix-7)	2/94400 (<0.01%)	0/47200 (0.0%)
XC7K480T (Kintex-7)	2/597200 (<0.01%)	0/298600 (0.0%)

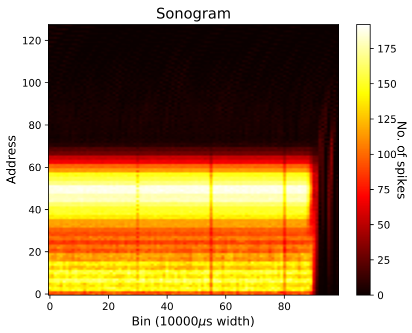
Secondly, four real-time tests were performed in order to analyze the impact of the spherical bushy cell module within the NAS in terms of number of output events and the NAS’ response in the frequency domain. The tests were organized as follows: 1) white noise sounds with the same volume (power equal to 0 dB); 2) white noise sounds with three different volumes (power equal to -6, 0, and 6 dB); 3) a set of pure tones with the same volume (amplitude equal to 1); and 4) a set of pure tones with three different volumes (amplitude equal 0.5, 1.0, 1.5, 2.0, and

2.5). The sounds were one second long, and the sample frequency was 48 MHz. The frequencies for the pure tones test were set to 100, 250, 500, 1000, 2000, 5000, 10000, 15000, 20000, and 25000 Hz.

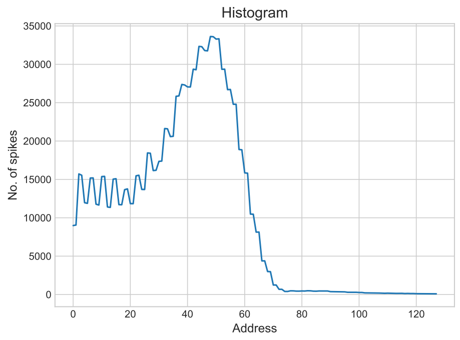
A 64-frequency-channel, cascade, mono NAS was generated using OpenNAS, and a set of 64 spherical bushy cells were also instantiated and connected to the NAS' output. This set of spherical bushy cell would create the AVCN module, which only contains as many spherical bushy cell modules as NAS frequency channels. The I2S-based input was used for sending the sounds from the computer to the FPGA-based board in real time. The output events were collected using the USBAERmini2 board and a Matlab script[3] for automating the process.



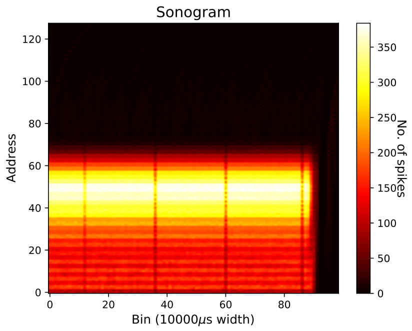
(A) 500 Hz, amplitude 0.5, histogram.



(B) 500 Hz, amplitude 0.5, sonogram.



(C) 500 Hz, amplitude 1.5, histogram.



(D) 500 Hz, amplitude 1.5, sonogram.

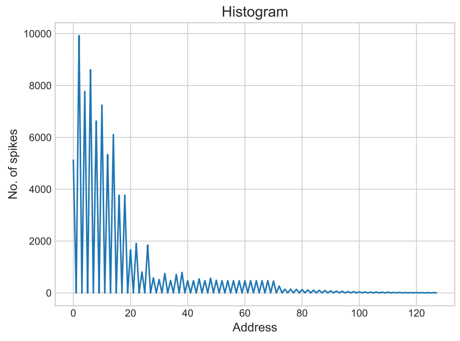
FIGURE 4.6: Histograms and sonograms for a 500 Hz pure tone with 0.5 and 1.5 of amplitude using a NAS without phase-lock.

The experiment was carried out twice by enabling and disabling the AVCN module for comparison purposes. Next, a few representative results among all

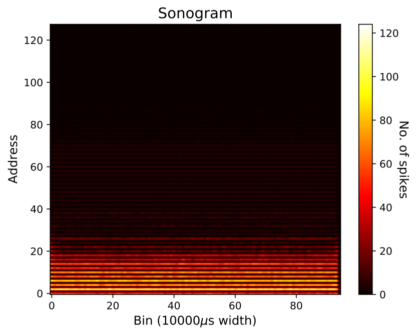
[3] https://github.com/dgutierrezATC/GenericSeqMon

the obtained outputs are shown. The full set of output results can be found on GitHub [4].

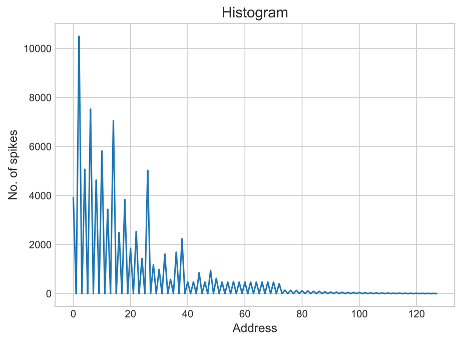
Fig. 4.6 shows the results obtained from the test in which the AVCN module was disabled, and a pure tone of 500 Hz was played at two different volumes (0.5 and 1.5). As it can be seen, both histograms have the main component around the same frequency. Fig. 4.7a presents a noise peak placed at high frequencies due to the low volume and the inherent error of the Spike-based Band-Pass Filter (SBPF) (Jimenez-Fernandez et al., 2017). Nevertheless, this error can be reduced while increasing the signal amplitude, as shown in Fig. 4.7c. Additionally, the number of events was different in both cases, having a maximum value of 17500 for amplitude 0.5 and 35000 for amplitude 1.5. These results matched the expected behavior of the NAS, which was already analyzed in (Jimenez-Fernandez et al., 2017; Dominguez-Morales et al., 2016).



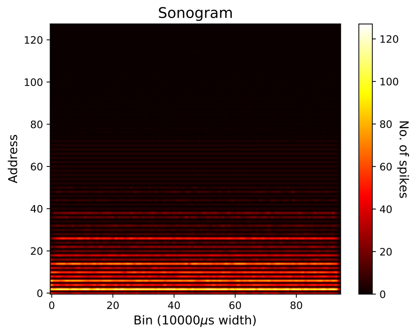
(A) 500 Hz, amplitude 0.5, histogram.



(B) 500 Hz, amplitude 0.5, sonogram.



(C) 500 Hz, amplitude 1.5, histogram.



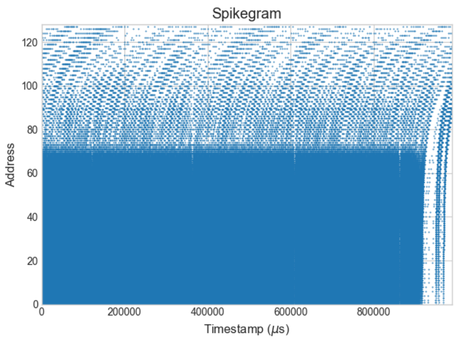
(D) 500 Hz, amplitude 1.5, sonogram.

FIGURE 4.7: Histograms and sonograms for a 500 Hz pure tone with 0.5 and 1.5 of amplitude using a NAS with the phase-lock module.

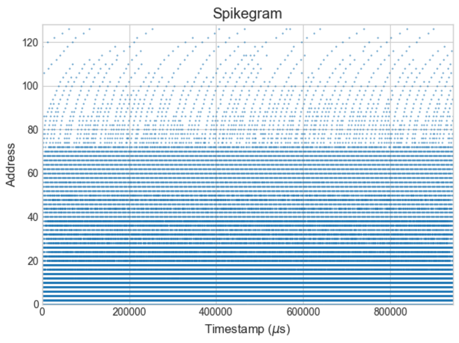
[4] In `https://github.com/dgutierrezATC/nssoc`, go to Examples, NAS_AVCN, NAS_I2S_64ch_mono_12att_monitor, realtest, GenericSeqMon, dataset

After the first experiment, which established the ground truth, the AVCN module was enabled and the whole experiment was carried out again. Fig. 4.7d shows the obtained results from the same cases shown in Fig. 4.6. Many differences can be appreciated from the results obtained in this experiment. For instance, there is only one peak in the histogram. However, this peak is not centered in the addresses that correspond to the pure tone frequency, but it is placed at the high frequency addresses. On the contrary, the addresses corresponding to the 500 Hz frequency showed an uniform response in terms of output events instead of a peak, as can be seen in Fig. 4.7a and Fig. 4.7c.

This effect is produced by the Spike-based Band-Pass Filter (SBPF) implementation since the band-pass filter is implemented by using two Spike-based Low-Pass Filter (SLPF) in serial for the NAS cascade architecture. Therefore, the achieved band is not as precise as desired, thus, having several frequency channels responding equally to the input sound (Jimenez-Fernandez et al., 2010; Jimenez-Fernandez et al., 2017). This way, the peak can be considered as noise, and its high value is also due to the tuning process of the SBPF, where the error is higher while the SBPF's middle frequency is also higher. Ideally, the output should present some activity around the addresses associated with the input sound and no activity in the rest.



(A) 500 Hz, amplitude 1.5, without phase-lock.



(B) 500 Hz, amplitude 1.5, with phase-lock.

FIGURE 4.8: Examples of recordings using NAS without and with phase-lock.

Furthermore, the volume of the input signal almost does not have any effect on the NAS output response. The maximum number of output events are practically identical. Nevertheless, two effects can be observed. Firstly, the flat part, i.e., the addresses with the same activity in the histogram, is shorter for louder volumes. Secondly, the peak's shape is softer when the volume is lower, and more abrupt when the volume is higher.

Related to the overall number of output spikes produced by the NAS, Fig. 4.8 shows the spikegram of both studied cases. Although the reduction of the activity is clearly visible, it was measured, obtaining 1314758 produced events when the AVCN module was disabled versus 83914 produced events when the AVCN module was enabled (a reduction of 1566%) for this case. This have some benefits, like lower power consumption. Also, a lower number of produced events means smaller FIFO memories and, therefore, less FPGA resources.

One of the negative aspects of the AVCN module is that the phase-locked spikes cannot be used for sound classification since both the frequency and the volume of the sound are needed for performing this task, nor for the ILD estimation in the LSO module, since the volume plays a key role. As an alternative, a mixed model including the zero-crossing detection and the ISI analysis could solve this problem since the information about the signal's amplitude will be encoded within the ISI value variation. In addition, spherical bushy cells can adapt themselves in order to tune the precise firing time according to the input sound (Kandel et al., 2000). This way, an adaptive version of the spherical bushy cell module would also be desired for future developments in the auditory attention field.

4.2.2 Implementing the Medial Superior Olive

Next, the MSO nucleus was modeled by also following the event-based approach presented by Jimenez-Fernandez et al. (Jimenez-Fernandez et al., 2010). The Jeffress model (Jeffress, 1948) was used as reference as well as the works presented in (Liu et al., 2013; Glackin et al., 2010). The implementation of the model was carried out by using a modular approach, since the MSO is basically composed by coincidence detector neurons and delay lines. Therefore, the basic modules were designed and implemented for being later encapsulated in other modules, thus, creating the ITD extraction network.

Lastly, the ITD extraction network module can be again encapsulated in a bigger module, instantiating as many modules as needed, for creating the MSO module. In this manner, the MSO module can be configured just by setting the number of frequency channels, the number of neurons per channel, and the maximum detection time. Thanks to the VHDL language features, this operation can be easily done by using the "*GENERIC*" statement for the configuration and the "*GENERATE*" statement for the automatic generation, thus avoiding setting the model manually.

4.2.2.1 Jeffress model implementation overview

Fig. 4.9 depicts the proposed model for implementing the ITD extraction based on the work of Jeffress (Jeffress, 1948). The original model was used due to its simplicity and as a proof-of-concept of an event-based ITD extraction model for FPGAs. This idea has been published in an international conference (Gutierrez-Galan et al., 2019b).

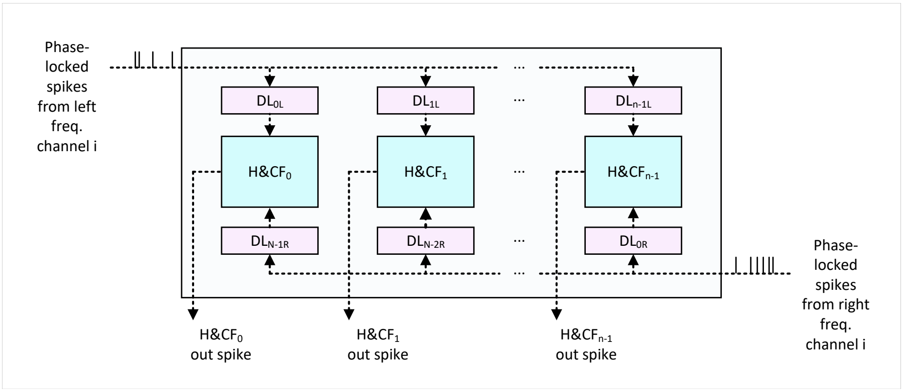


FIGURE 4.9: Block diagram of the proposed implementation for the Jeffress model.

As it was mentioned before, the model is composed of two main modules: 1) the coincidence detector neuron (H&CF), represented in Fig. 4.9 by the blue box, and 2) the delay line (DL), represented in Fig. 4.9 by the pink box. It has two inputs, the *"left_input_spikes"* and the *"right_input_spikes"*, which correspond to the phase-locked NAS' output from the i-th left and right frequency channels, respectively. The number of outputs coincides with the number of coincidence detector neurons, and it is set by the generic parameter *NUM_NEURONS*.

In addition, the number of delay line modules will be twice the number of neurons since each coincidence detector neuron receives inputs from both left and right channels. Next subsections will describe each module in detail, as well as showing the results from both simulations and real-time tests.

4.2.2.2 Coincidence detector neuron model

The LIF neuron was mostly used for modelling the coincidence detector neuron (Liu et al., 2013; Glackin et al., 2010). The coincidence detection time can be established by finely tuning the neuron's parameters or even by applying learning procedures as Spike-Timing-Dependent Plasticity (STDP) (Glackin et al., 2010). In our case, a simplified version of the LIF neuron called SHF and proposed by Jimenez-Fernanez et al. (Jimenez-Fernandez et al., 2010) was used. The original model has two inputs, called *positive_input* and *negative_input*.

At the initial state, when an input event is received, it is internally held during a specific time. If no events arrive during that time period, the held event is released and put in the output. However, if an event is received while the event is held, the output will depend on both the held event and the incoming event. For instance, if a positive spike is held and another positive spike is received, the module will produce an output positive spike, and a positive spike will be held. This way, both the excitatory and inhibitory synapses were modelled, as well as the temporal decay of the membrane potential. The complete behavioral description of the SHF model can be found at (Jiménez Fernández, 2010).

Although it is proven that the LIF neurons in the MSO have both excitatory and inhibitory projections, the simplified version of the Jeffress model (Jeffress, 1948) only considered the excitatory projections. Moreover, a coincidence can be detected just with one spike from each side without the need of integrating more spikes during the coincidence detection time due to the phase-locked spikes.

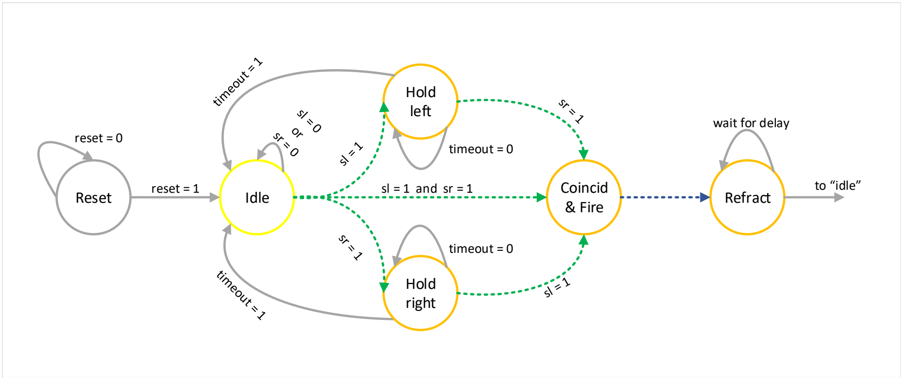


FIGURE 4.10: FSM of the Spike Hold&Coincidence Fire (SHCF) module.

Based on the SHF module, the Spike Hold&Coincidence Fire (SHCF) module was proposed in order to adapt the original model to the coincidence detector neuron requirements. This module also has two inputs, but in this case both of them are positive inputs: one for the left phase-locked spike (*"left_spike"*) and one for the right phase-locked spike ((*"right_spike"*)). Finally, it has one output (*"coincidence_output_spike"*), that represents the output spike when a coincidence is detected. The detection time is set by a generic parameter (*TEMPORAL_COINCIDENCE_WINDOW*), expressed in microseconds. Fig. 4.10 shows the Finite State Machine (FSM) that implements the coincidence detection behavior.

Initially, the SHCF module is in *IDLE* state, and it remains there until an input spike is detected. When an input spike is received, the next state is either *HOLD_LEFT* or *HOLD_RIGHT*, depending on the input source. Then, the module stays in this state until either the coincidence time finishes or if a spike from the contralateral side is received. If no contralateral spike is received, the next state is again *IDLE*, meaning that no coincidence was detected. If the input spike is coming from the ipsilateral side, it is ignored. This case should not occur due to the phase-locked spikes prevent from this effect. Nevertheless, if that happens, it would mean that the detection time configuration is not valid.

Finally, if a contrary side spike is received while being at *HOLD_LEFT* or *HOLD_RIGHT* state, the next state is set to *COINCIDENCE_AND_FIRE*, producing an output spike. This state can also be reached if both left and right spikes are received at the same clock cycle while in *IDLE*. After producing the output spike, the module goes into the *REFRACTORY* state for one clock cycle, and then to *IDLE* again.

TABLE 4.2: Hardware resources utilization of the SHCF model with detection time of 50 μs for different FPGA devices.

FPGA chip	Slice Registers Used / Available	Slice LUTs Used / Available
XC6SLX150T (Spartan-6)	15/184304 (0.01%)	27/92152 (0.03%)
XC7A75T (Artix-7)	15/94400 (0.02%)	27/47200 (0.06%)
XC7K480T (Kintex-7)	15/597200 (<0.01%)	27/298600 (0.01%)

The VHDL implementation of this module was done by implementing a FSM-based architecture. A behavioral simulation, shown in Fig. 4.11 was carried out in order to verify the correct functionality of the model, and a resources consumption report, summarized in Table 4.2, was generated. As it can be observed, a single SHCF unit needs low resources, thus allowing to deploy many units in a single FPGA chip for implementing large models.

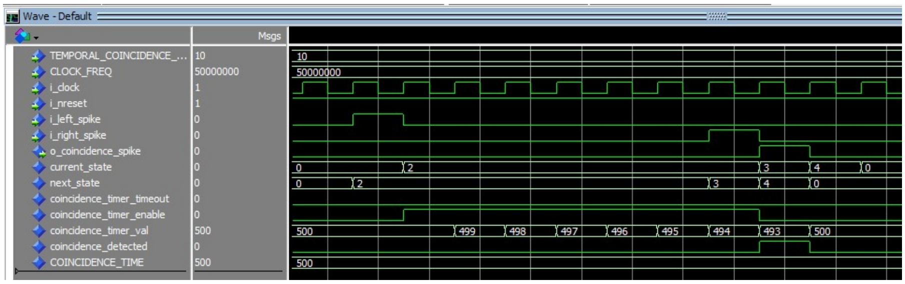


FIGURE 4.11: Simulation results of the SHCF module. States 0, 2, 3, and 4 corresponds to *IDLE*, *WAIT_RIGHT*, *COINCIDENCE*, and *REFRACTORY*, respectively.

4.2.2.3 Delay line model

The delay line, or transmission line, is the axon of a spherical bushy cell that projects into the coincidence detector neuron. For the ITD extraction, the axonal delays are tuned to transmit the spikes with a certain delay and, therefore, create the effect described by Jeffress. In addition, the axon's length affects the axonal delay, being this feature fixed and not adaptive.

From the digital design point of view, it can be implemented as a timer, where the input enables the timer, and the output is the interruption that notifies the end of the timer. Nevertheless, a simplified version of the SHF was used. In this case, the module only has one input, *"spike_in"*, and one output, *"spike_delayed"*. Furthermore, the transmission time is configured by the generic parameter *TRANSMISSION_TIME*, expressed in microseconds.

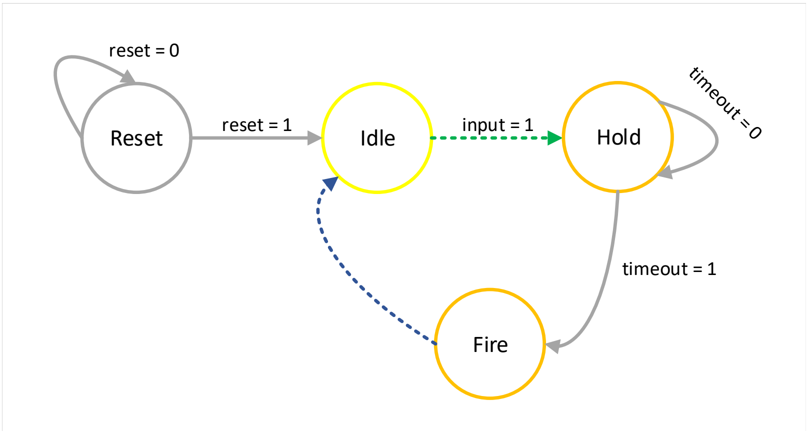
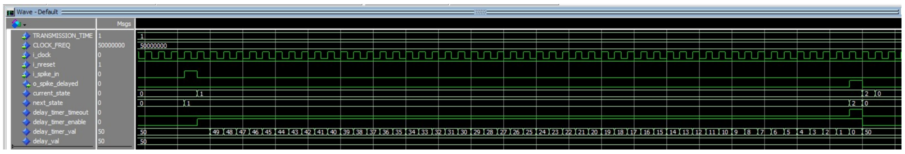


FIGURE 4.12: FSM of the Delay Line module.

Fig. 4.12 shows the FSM of the delay line module. Its behavior is similar to the SHF's FSM. By default, the module is in *IDLE* state, and stays on it until an input spike is received. Then, the next state is *HOLD*, where the module will stay during a time period equal to *TRANSMISSION_TIME*. While this state is active, if a new input spike is received, it is not taken into account, thus being discarded. After the timer ends, the held spike is released, passing to the *FIRE* state. Finally, the module goes again to the state *IDLE*, where it will be waiting for the next input spike. Fig. 4.13 shows a simulation screenshot of the DL module.



```
TRANSMISSION_TIME    1
CLOCK_FREQ           50000000
i_clock              0
i_nreset             1
i_spike_in           0
o_spike_delayed      0
current_state        0
next_state           0
delay_timer_timeout  0
delay_timer_enable   0
delay_timer_val      50
delay_val            50
```

FIGURE 4.13: Simulation results of the DL module. States 0, 1, and 2 to *IDLE*, *HOLD*, and *FIRE*, respectively.

In theory, a new input event should not arrive to the module while being in the *HOLD* state due to the phase lock module, as it was mentioned before. Therefore, this case was not taken into account in the delay line module design. As future work, either a FIFO memory or an accumulative timer could be used for collecting all the input events without discarding any of them at the expense of increasing the resources consumption of the module. Table 4.3 summarizes the resources consumption of the delay line module.

TABLE 4.3: Hardware resources utilization of the delay line model with transmission time of 700 μs for different FPGA devices.

FPGA chip	Slice Registers Used / Available	Slice LUTs Used / Available
XC6SLX150T (Spartan-6)	19/184304 (0.01%)	29/92152 (0.03%)
XC7A75T (Artix-7)	17/94400 (0.02%)	30/47200 (0.06%)
XC7K480T (Kintex-7)	17/597200 (<0.01%)	30/298600 (0.01%)

Compared to the SHCF module, the delay line module requires a few more resources even though its FSM has less states. This is because the internal timer of the delay time module works with larger time periods, thus needing more bits to implement the timer.

4.2.2.4 ITD extraction network model

After the two main components of the Jeffress model were implemented, the coincidence detector network for extracting the ITD was designed. As it was mentioned in Section 4.2.2.1, the network can be easily deployed by instantiating the same two modules as many times as desired. Fig. 4.9 shows the block diagram of an ITD extraction network with n coincidence detector neurons (or SHCF) modules. For each SHCF module, two delay line modules are needed: one for the left spike and one for the right spike, thus having $2n$ delay line modules.

The ITD extraction network module has two inputs: *"left_channel_spike"* and *"right_channel_spike"*, that corresponds to the spikes coming from the left and right spherical bushy cell associated to one specific frequency channel of the NAS. In addition, it has as many outputs as the number of coincidence detector neurons specified in the generic parameter *NUM_NEURONS*. The SHCF's ID represents the spatial localization of the detected sound source in such a way that the coincidence detector neuron with ID equal to 0 will receive first the events coming from the left cochlea (left delay line with transmission time equal to zero) and lately the events coming from the right cochlea (right delay line with transmission time equal to *MAX_DETECTION_TIME*). This way, this neuron will only fire a spike when the sound source is placed completely on the right, meaning that the sound will first reach the right ear and then, after 700 μs approximately, the left ear.

The maximum ITD that the network is able to detect can be configured through the generic parameter *MAX_DETECTION_TIME*, expressed in microseconds. Furthermore, based on the work presented in (Liu et al., 2013), an overlapping time parameter, called *DETECTION_OVERLAP*, was added. This overlap increases the coincidence detection accuracy by producing, in some cases, two coincidence spikes at the output.

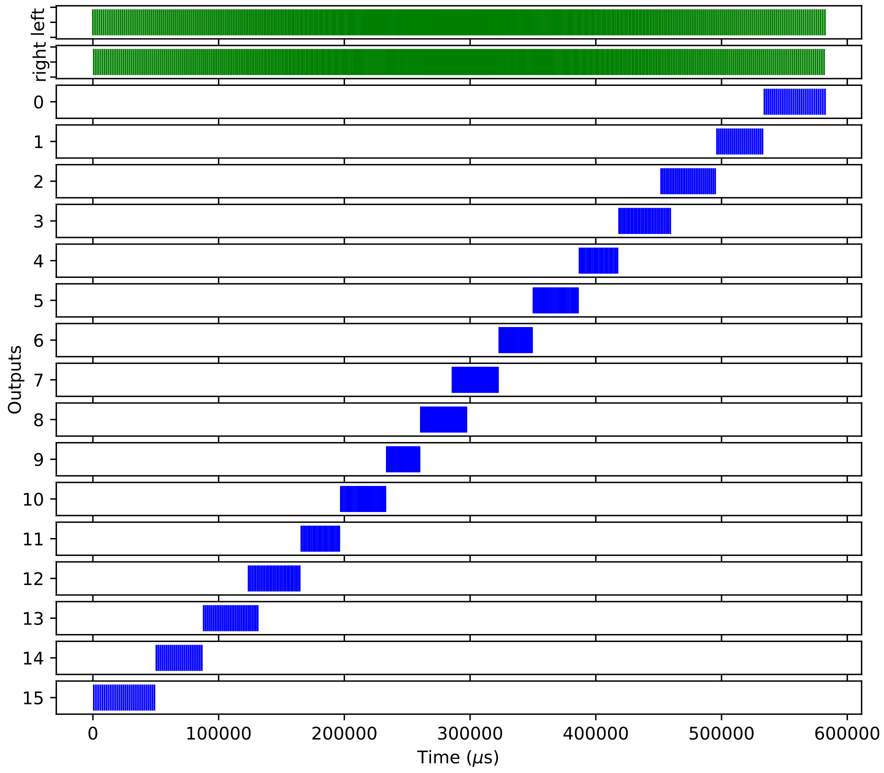


FIGURE 4.14: Example of the ITD network when using a sweep.

Therefore, the detection time bin of a single coincidence detector neuron depends on the global detection time of the network, as well as the total number of coincidence detector neurons deployed, and the detection overlap. It can be defined as in Equation 4.1.

$$detection\_time\_bin = \frac{MAX\_DETECTION\_TIME}{NUM\_NEURONS} + DETECTION\_OVERLAP \tag{4.1}$$

In the same way, the transmission time of a single delay line is calculated also based on the global detection time of the network, as well as the total number of coincidence detector neurons deployed, and it is proportional to the index of that specific transmission line. For instance, the first delay line (with index equal to 0) has no delay, the second one (with index equal to 1) has one time the

$\frac{MAX\_DETECTION\_TIME}{NUM\_NEURONS}$, and so on. Thus, the transmission time of the i-th delay line can be defined as in Equation 4.2.

$$transmission\_time_i = \frac{MAX\_DETECTION\_TIME}{NUM\_NEURONS} \times i \tag{4.2}$$

A behavioral simulation was carried out for validating the module. The test consisted of the generation of an input spike pair where the time difference between them was increasing over time. Therefore, a sound source moving from left to right with respect to the reference can be simulated. The *NUM_NEURONS* parameter was set to 16, the *MAX_DETECTION_TIME* parameter was set to 700 μs, and the *DETECTION_OVERLAP* parameter was set to 5μs.

The generated input stimuli were organized in bursts of five spike pairs with one millisecond between pair generations. Firstly, the time difference between the left and right stimulus was set to 700 μs, meaning that the sound source was completely placed at the left with respect the reference, creating a 90 degree angle. Then, for each new spike burst, the time difference value was decreased by 20 μs, simulating the movement of the sound source towards the center with respect to the reference. This process was repeated while the time difference was higher than zero, meaning that the sound source was not placed in front of the reference, where the time difference is zero.

Secondly, once the simulated sound source was placed in front of the reference, with an angle of zero degrees, the time difference was increased for each burst again by 20 microseconds but producing first the right spike and then the left spike, simulating that the sound source was being moved towards the right ear. The simulation was stopped when the time difference reached 700 μs, meaning the sound source was then placed at a 90 degree angle on the right position with respect to the reference. Results from the simulation are shown in Fig. 4.14.

Two main effects can be clearly observed. On the one hand, it can be seen that the burst's width decreases when the time differences are closer to zero since the coincidence is detected some microseconds earlier in these cases compared to high time differences. On the other hand, there are some cases where two consecutive neurons detected a coincidence produced by the same input stimulus pair due to the overlapping configuration. In general, the network worked according to the expected output, proving that the model was successfully implemented.

A resources consumption report of this model was generated for different FPGA chips and collected in Table 4.4. For a single ITD detection network, 2.89% of the resources are needed in the worst case, meaning that a total of 34 ITD detection networks could be deployed without taking into account the NAS module. Since the ITD extraction is valid for low frequency channels, a set of ten

TABLE 4.4: Hardware resources utilization of the ITD network model with a global detection time of 700 μs and 16 coincidence detection neurons for different FPGA devices.

FPGA chip	Slice Registers Used / Available	Slice LUTs Used / Available
XC6SLX150T (Spartan-6)	788/184304 (0.42%)	1294/92152 (1.40%)
XC7A75T (Artix-7)	764/94400 (0.81%)	1364/47200 (2.89%)
XC7K480T (Kintex-7)	764/597200 (0.12%)	1364/298600 (0.45%)

to fifteen ITD detection networks could be enough, thus still having free resources for future implementations.

4.2.2.5 Medial Superior Olive model

Based on the biological principles of the MSO introduced in Section 1, it can be modelled as a set of ITD detection networks tonotopically organized and connected one-to-one to the phase-locked NAS output spikes from low-frequency channels. In this manner, a MSO module was implemented for automatically deploying as many ITD detection networks as specified by the generic parameters, thus facilitating the modules' configuration and the output events collection. As input lines, the module has four times the value of the generic parameter *NUM_FREQ_CH*, since for each frequency channel it will have inputs for both left and right, as well as positive and negative spikes for each side. As output, the module has an AER interface, thus allowing the communication with an external device or with other modules within the same FPGA.

The model was composed by layers, where each layer consisted of two spherical bushy cell modules (SBC), an ITD detection network module, and a local events monitor module, as shown in Fig. 4.15. The number of layers can be defined by the generic parameter *NUM_FREQ_CH*, that indicates the number of NAS' frequency channels from which the events will be received. Two generic parameters, *START_FREQ_CH* and *END_FREQ_CH*, were also needed in order to set both the initial and the final NAS frequency channels that are going to be used. In addition, these values were also used to configure the channel's address for each local monitor.

In the same way that the ITD detection network module, the networks' configuration were carried out through the generic parameters *NUM_NET_NEURONS*, *MAX_DETECTION_TIME*, and *DETECTION_OVERLAPING*. These parameters are shared by all the ITD detection networks inside the MSO module, and they cannot be modified in

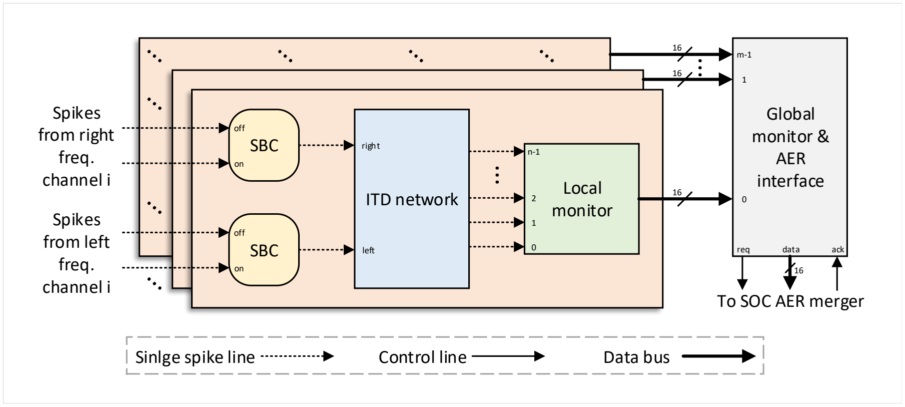


FIGURE 4.15: Block diagram of the proposed MSO model.

real-time. This feature could be implemented by adding a dedicated Random Access Memory (RAM) memory which would be modified by an external controller, and it will be implemented in the next version.

For collecting the events from each ITD detection network, a spikes monitor was used. In a study carried out by Cerezuela-Escudero et al. in (Cerezuela-Escudero et al., 2013), where different approaches for implementing an event monitor were compared, the global monitor was identified as a potential bottleneck point when the output spike rate was high enough. In addition, both the organization and the number of spike lines in the module which is being monitored were also considered as a critical aspect when designing a spike monitor. The work compared a global spikes monitor, where all the spikes were sent to a single monitor, with a distributed spikes monitor, where the spikes where divided in groups and, for each group, a local monitor was used. This way, the workload was distributed, thus reducing the bottleneck problem of the global monitor output AER interface. Results showed that the distributed monitor was more efficient in terms of events loss, as well as in terms of FPGA resources consumption.

Therefore, a distributed events monitor, composed of several local monitors and a global monitor, was used in this work. The block diagram of the local monitor module is shown in Fig. 4.16. First, an event encoder was used for identifying the input spike and labelling it with a number between $[0, NUM\_NET\_NEURONS - 1]$. The event encoder module was designed in such a way that only one event is encoded at each clock cycle. If two or more events arrive at the same time, the event with the lowest ID (i.e., closer to zero) will be encoded.

Then, the encoded events are stored in a FIFO memory. The reading process is controlled by the arbiter module in the global monitor module. When the

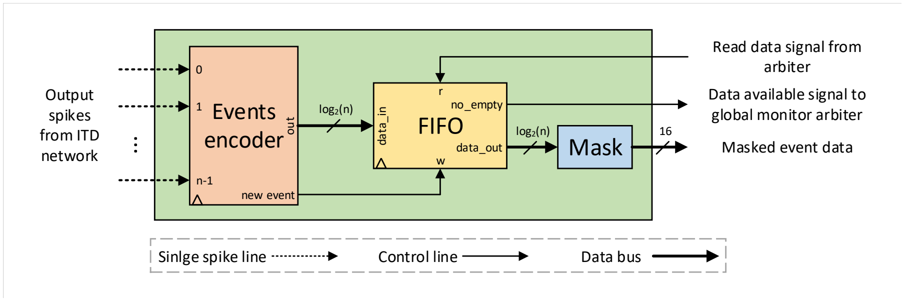


FIGURE 4.16: Block diagram of the MSO local events monitor.

arbiter enables the read signal, a data from the FIFO is read, and a mask is applied to the read data. This mask module concatenates the already stored neuron label with:

- Event polarity, that is always zero.
- Associated NAS frequency channel, determined by the ID of ITD detection network within the MSO module and both *START_FREQ_CH* and *END_FREQ_CH* parameters.
- Left/right ear, which is always zero since only one MSO nucleus is instantiated.
- One bit fixed to '0' for indicating that the event was produced by the MSO module.
- One bit fixed to '1' for indicating that the event was produced by the SOC module and not by the NAS.

The resultant masked event is then sent to the global monitor module, where it is stored in another FIFO memory. The arbiter's output is used as selector for a multiplexer, which selects the correct masked event data to be stored in the FIFO memory. Therefore, this memory will contain data from all the ITD detection networks. Finally, the data stored in the FIFO memory is sent out through the AER out interface to either another FPGA module or an external device. The block diagram of the global monitor module is shown in Fig. 4.17.

As it was done before, a resources consumption report of this model was generated but only for the Artix-7 FPGA chip, which will be used for the real-time experiments. Table 4.5 shows the obtained results for a MSO module with 10 ITD detection networks (connected to the NAS frequency channels between 25 and 34), 16 coincidence detection neurons for each ITD network, and a global detection time of 700 μs with an overlapping time of 5 μs. As it can be seen, the MSO module almost needs 30% of the FPGA's Look-Up Tables (LUTs) due to the multiple instantiations of both the delay lines and the coincidence detector

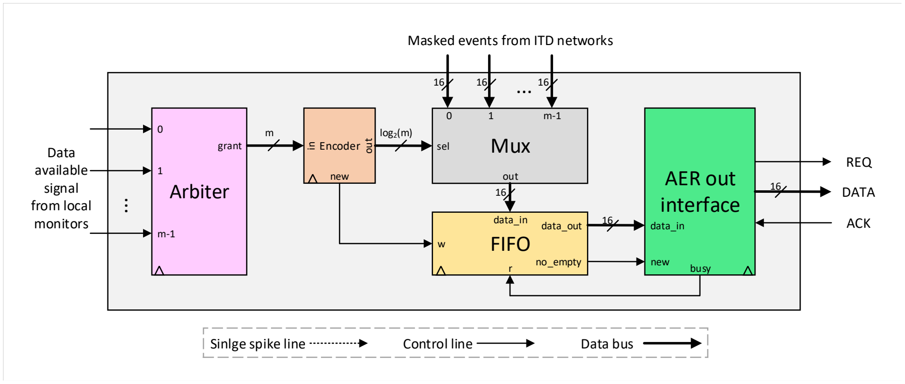


FIGURE 4.17: Block diagram of the MSO global events monitor.

neurons. This means that for a more complex MSO model, an FPGA with much more resources would be needed, taking also into account that there should be at least 50% of the resources available for the NAS model.

TABLE 4.5: Hardware resources utilization of the MSO model with a global detection time of 700 μs, 10 ITD networks and 16 coincidence detection neurons per ITD network for the Artix-7 FPGA chip.

Module name	Slice Registers Used / Available	Slice LUTs Used / Available
Left AVCN	20/94400 (0.02%)	0/47200 (0.0%)
Right AVCN	20/94400 (0.02%)	0/47200 (0.0%)
MSO	7640/94400 (8.09%)	13649/47200 (28.92%)
MSO monitor	229/94400 (0.24%)	331/47200 (0.70%)

Once again, a behavioral simulation was performed in order to validate the implemented module. In this case, the input stimulus was not generated manually due to its complexity. Instead, a NAS model was used for generating an events file. Firstly, a 64-frequency channels, stereo NAS with I2S-based input interface was generated. Then, a virtual room with a pair of microphones and a sound source was created by using the Room Impulse Response (RIR) generator (Habets, 2006) Python package. The room's dimensions were 10 $\times$ 10 $\times$ 4 meters (x, y, and z). The microphone pair (subcardioid type, oriented as the human ears, with a separation of 0.3 meters) was placed at (5, 3, 2), and the sound source was placed at (5, 4.5, 2). The sound velocity was set to 340 m/s, and the reverberation time was set to 0.2 seconds. Finally, the sample frequency was set to 48000 Hz, and a pure tone of 1000 Hz with 0.5 seconds of duration was played.

The output audio file, generated by the RIR generator tool, was then used as input for the NAS. The NAS was deployed into the ZTEX 2.13 board along with two custom boards: a baseboard (see Appendix C.2), and an audio input board (see Appendix C.1) for interfacing the baseboard with the computer audio interface. The audio file was played from the computer with a 80% of volume, and the output events were collected into an .aedat file by using the USBAERmini2 board and jAER (Delbruck, 2007).

Next, the events file was loaded by using a Python script and pyNAVIS tool (Dominguez-Morales et al., 2021b). After checking that all the values were correct according to the settings, two lists were generated: one containing the addresses and another one containing the timestamps. Those lists were used as input files for the VHDL simulation, and the output events generated by the MSO model were also saved as a text file with a specific format. Finally, the results files were loaded by using pyNAVIS[5], and a set of plots were generated. These plots are shown in Fig. 4.18.

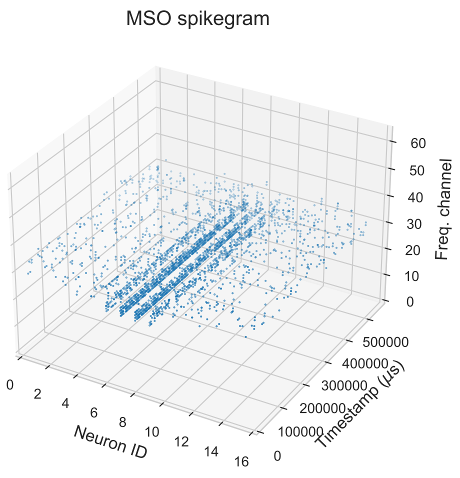
On the top left, the MSO spikegram is shown. This plot shows the raster plot, or the raw output events, produced by the MSO module. The x-axis represents the neurons' IDs, the y-axis represents the time, and the z-axis represents the NAS frequency channels, being channel 0 the one with the highest frequency and channel 63 the one with the lowest frequency. Since the sound source was placed in front of the microphone pair, the main activity was produced by neurons with IDs between 6 and 10 along the simulation. However, some activity was generated by other neurons which do not correspond to the correct localization due to the noise added when using a real-time setup for generating the recording.

On the top right, the MSO histogram is presented. The histogram shows the same information that the spikegram but without taking into account the time. Instead, the total number of events generated by each neuron in each frequency channel is shown. As it was mentioned before, the main activity was found in the neurons placed at the middle. Nevertheless, not all the frequency channels presented activity in those neurons. Frequency channels with a middle frequency close to the pure tone frequency had a better response compared to those frequency channels that have lower middle frequencies.

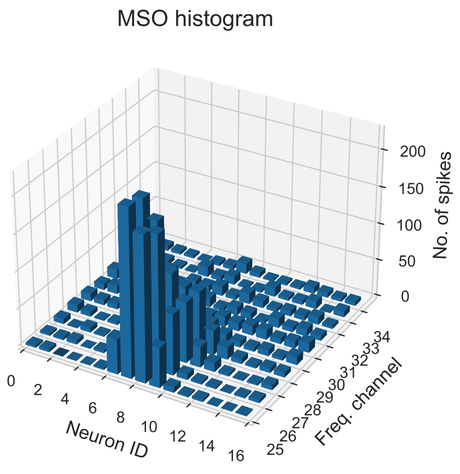
On the bottom left, the MSO heatmap is shown. In this plot, the same information of the histogram is shown but just using a two dimensions color map: the frequency channels and the neuron IDs. The number of output events is represented by a color code, where the meaning of each color is determined by the color bar placed on the left. In addition, the color assigned to each position is calculated relatively to the maximum value. Therefore, this view facilitates the identification of the main activity within the MSO model and gives an idea on how sparse was the model response to the input stimulus.

Finally, on the bottom right, the position estimation extracted from the MSO output is plotted. For generating this plot, a software version of the coincidence

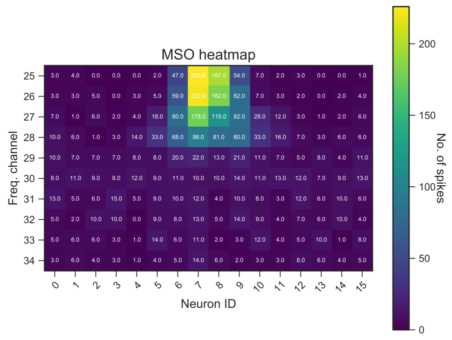
[5]The functionalities needed for loading, verifying, postprocessing and plotting the MSO model information were implemented as part of this thesis.



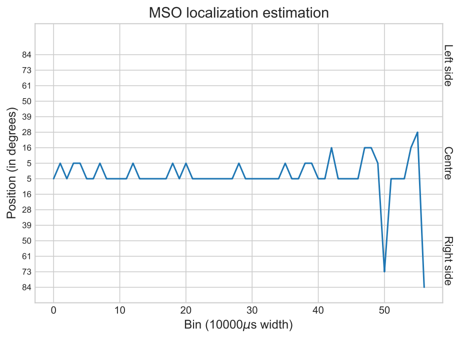
(A) MSO spikegram.



(B) MSO histogram.



(C) MSO heatmap.



(D) MSO localization estimation.

FIGURE 4.18: Results from a behavioral simulation of a MSO model when using a 1000 Hz pure tone, placed in front of the reference, with a distance of 1.5 meters. The plots were generated using pyNAVIS.

counters from the Jeffress model was used. First, a time bin is defined for integrating the events of each coincidence detector neuron. Then, the events are counted, and the neuron with the maximum fired events is selected as the winner. Finally, the ID of the winner neuron is converted to an angle, being this angle the estimated position of the sound source in the horizontal plane with respect to the microphone pair, as shown in Equation 4.3. The value of *max_angle_accuracy* can be obtained according to Equation 4.4, and it directly depends on the model's architecture.

$$estimated\_angle = max\_angle\_accuracy \times neuron\_ID \tag{4.3}$$

In this case, it can be seen that the position of the sound source was estimated in a range of ±5 degrees with respect to the reference, which is in front of it. Since there was an even number of coincidence detector neurons, the winner neuron was oscillating between two consecutive positions. The big oscillations observed at the end of the plot correspond to the end of the input stimulus, where the MSO model did not receive any sound, thus producing a random output due to the existing background noise.

The accuracy of the model directly depends on the number of coincidence detector neurons and it could be measured from this plot. However, the accuracy also depends on many other factors, such as the room shape and its dimensions, the room's reverberation, the input sound, the distance between microphones and the sound source, etc. Therefore, the accuracy study can be considered application-specific measurements, thus not being carried out in this experiment.

This process was not integrated along with the MSO model since the time bin depends on the final application, thus needing to be configured or even adapted in real time. In addition, most of the times the MSO events will be sent to a neuromorphic computing platform, like SpiNNaker, for further processing like sensory integration, multiple sound sources localization, or auditory attention. Nevertheless, the coincidence counter module was implemented and tested using VHDL for future works.

4.2.3 Lateral Superior Olive

According to the block diagram showed in Fig. 4.2, the next nucleus to be implemented would be the LSO for the ILD extraction. A reduced event-based model of the LSO was already implemented by Cerezuela-Escudero et al. in (Cerezuela-Escudero et al., 2018). In that work, the SHF model was used and combined in groups of four to create a new module: the spike firing rate subtractor. This module is able to produce a spike stream whose firing rate is proportional to the rate difference of two input spike streams. Therefore, the ILD is encoded in the output spike rate. Details about the implementation can be also found in (Jiménez Fernández, 2010).

The basic module has four inputs: two of them are for the spikes coming from the left cochlea (positive and negative spikes), being considered as the excitatory input; and the other two inputs are for the spikes coming from the right cochlea, being considered as the inhibitory input. Since the NAS model already implements positive and negative spikes, a model of the Medial Nucleus of the Trapezoid Body (MNTB) nucleus was not needed.

Therefore, according to the design strategy used for the implementation of the MSO module, the LSO module would be implemented for automatically generating a set of SHF modules. The number of ILD extraction units would be equal to the number of frequency channels specified in the generic parameter

NUM_FREQ_CHANNELS that indicates the number of frequency channels from which the ILD will be extracted. In addition, the module would also have two generic parameters for indicating both the first and last frequency channel addresses (*START_FREQ_CH* and *END_FREQ_CH*). Finally, the number of input signals would be four times the value of *NUM_FREQ_CHANNELS* since, for each ILD extraction unit, two inputs from the left cochlea and two inputs from the right cochlea are needed. The number of output signals would be twice the value of *NUM_FREQ_CHANNELS* since, for each ILD extraction unit, both positive and negative output signals are available.

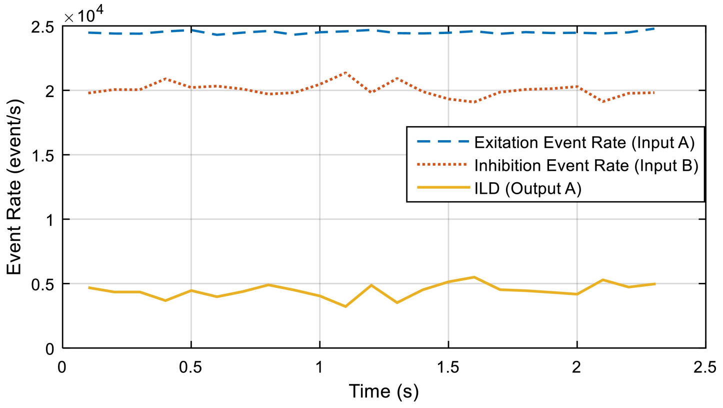


FIGURE 4.19: ILD (encoded as an event-rate) measured over time obtained from the input data that generates the left and right NAS when the sound source is placed at 45° from the head. Figure taken from (Cerezuela-Escudero et al., 2018).

Fig. 4.19 shows an example of the LSO model response. A sound was played with an angle of 45 degrees with respect to the head. As it can be seen, the output firing rate of input A was higher than the output firing rate of input B since the sound source was placed closer to the input A, thus receiving the sound earlier and with more power than to the input B. In fact, the event rate of input A is $2.5 \times 10^4 events/sec$, while the event rate of input B is $2 \times 10^4 events/sec$, thus obtaining an estimated ILD of $0.5 \times 10^4 events/sec$.

According to the biological principles described in Chapter 1, the ILD works better with high frequency sounds. The ILD estimation can be used for performing the sound source localization tasks, but it is also used for helping the MSO to better determine the position of the sound source by correcting the ambiguity that the high frequency sounds produce at the MSO (Dávila-Chacón et al., 2018).

Nevertheless, the latest research about the LSO and the ILD extraction suggests that the LSO's neurons are temporal differentiators rather than integrators (Franken et al., 2018), meaning that those neurons not only take into account the firing rate but also the precise timing of the input stimuli, thus acting similar to the MSO. Therefore, further research on this aspect is needed for implementing a reliable model of the LSO.

In this thesis, new implementations of the LSO model were not carried out. Instead, the model proposed by Cerezuela-Escudero et al. will be used in the future as part of the SOC model proposed in this thesis. As it was mentioned before, the combination of the MSO and LSO outputs will be performed by a neuromorphic computing platform such as SpiNNaker or Loihi due to the complexity of the SNN model for implementing the Inferior Colliculus (IC), which is the nucleus where this process is carried out.

4.2.4 Implementing the Superior Olivary Complex

The next design step was the implementation of the SOC module. It was implemented by using the modules detailed in previous sections. Therefore, the proposed SOC module was composed by two AVCN modules, a MSO module with an events monitor, the LSO module also with an events monitor, a Read Only Memory (ROM) memory for storing the configuration parameters, and an AER merger module for collecting AER events that come from both monitors and send them out through a single AER output interface. The block diagram of the proposed model is depicted in Fig. 4.20.

As input, the SOC module receives the raw output spikes from the NAS frequency channels. The frequency channel addresses, as well as the number of frequency channels, are defined by the configuration parameters stored in the ROM memory, which were already described in previous sections. As output, the module has an AER interface for sending the events either to an another module or to an external device. Internally, the AER merger receives the AER events from both the MSO and LSO monitors, and it works as an AER-based arbiter. More details about this module can be found in (Rios-Navarro et al., 2016).

With this SOC module, both binaural cues can be extracted in parallel from the same input data. In addition, each submodule inside the SOC module can be adapted according to the application due to the hierarchical design strategy followed during the whole implementation process. It will facilitate the implementation of new features without the need of modifying input/output interfaces, as well as to disable unused modules for saving both power and FPGA resources.

4.2.5 Integrating NAS and SOC: The Neuromorphic Auditory Complex

Finally, the integration of the SOC module with the NAS module was carried out. Similar to the implementation of the SOC module described in Section 4.2.4, and taking into account the block diagram shown in Fig. 4.2, three modules

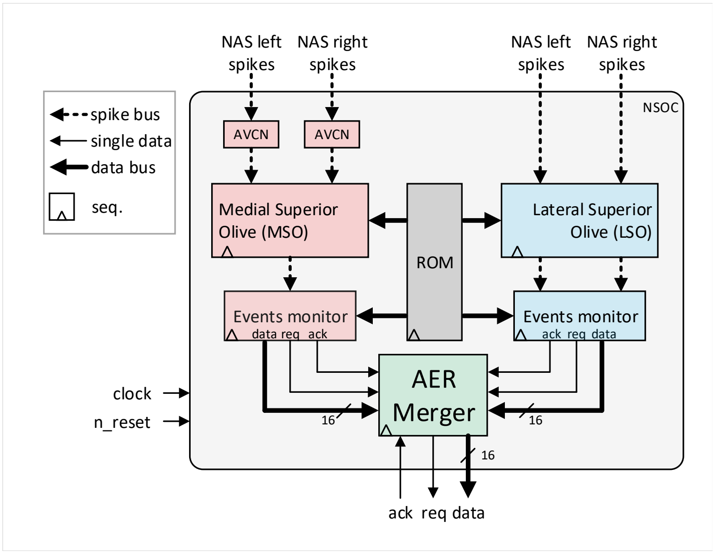


FIGURE 4.20: Block diagram of the proposed SOC module.

were instantiated within a top module, called Neuromorphic Auditory Complex (NAC). Fig. 4.21 shows the block diagram of the NAC module.

Firstly, a stereo NAS model was added. The stereo NAS takes the role of both cochleas. As it was already explained, the NAS configuration is carried out when it is generated by OpenNAS tool (see Section 3.2). Therefore, no memories are needed for storing the configuration values. Secondly, the SOC module was added. As it was mentioned before, only the MSO module was implemented, although the whole system was already prepared for having both the MSO and the LSO models. Lastly, the AER merger module was added for collecting the output events from both previous modules and sending them out to either another module or an external device.

The NAC module can have two different input interfaces, like the NAS. These interfaces are 1) I2S-based interface and 2) PDM microphones. The I2S interface was selected by default since it allows to connect the ZTEX 2.13 FPGA-based board with the computer through an audio cable for playing sounds. The NAC only has one output interface: the AER interface. No other interfaces were implemented since most of the neuromorphic computing systems use this AER interface as standard for communicating with neuromorphic sensors. Therefore, the NAC could be interfaced directly with SpiNNaker [6] and with Loihi.

[6]The communication with the SpiNNaker board is carried out through a custom Verilog module implemented by the SpiNNaker team. A modified version of this module for supporting the NAC output package format can be found in `https://github.com/dgutierrezATC/NAS_SpiNNaker_interface`.

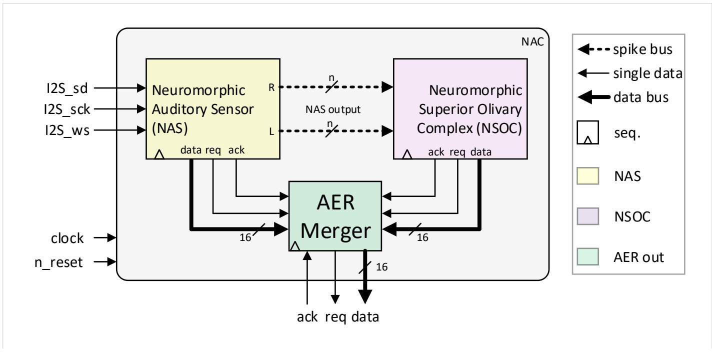


FIGURE 4.21: Block diagram for the integration between the NAS model and the SOC model.

As it can be seen in Fig. 4.21, the NAS's output is sent to both the SOC module and the AER merger module in a different way. NAS's raw output events, that corresponds to the Spike-based Band-Pass Filter (SBPF)'s output, are sent to the SOC module, while NAS's AER events are sent to the AER merger. Furthermore, SOC's output events are also sent to the AER merger. The format of the AER event package is shared by both modules, and it is shown in Fig. 4.22.

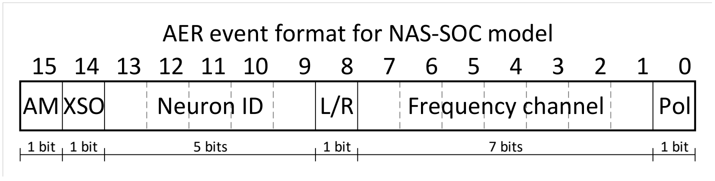


FIGURE 4.22: Event package format.

This format was designed based on the NAS original AER event format[7] in order to maintain the compatibility with the already existing NAS-based applications. Each package contains 16 bits of information organized in 6 different fields, which are described next:

- Bit 15: auditory model ID (AM). This bit is used for identifying if the event was produced either by the NAS module ('0') or by the SOC module ('1').
- Bit 14: SOC model ID (XSO). This bit is used for identifying if the event was produced either by the MSO model ('0') or by the LSO model ('1') within the SOC module. This bit is taken into account only if the AM bit is set to '1'.

[7]See https://github.com/jpdominguez/NAVIS-Tool/wiki/Software-architecture for more information about the NAS AER events format.

- Bits 13 to 9: neuron ID (NID). This 5 bits are used for identifying the neuron ID that produced the event either from the MSO or the LSO. Therefore, a maximum of 32 neurons for each frequency channel is allowed. These bits are taken into account only if the AM bit is set to '1'.
- Bit 8: left/right cochlea (L/R). This bit is used for indicating if the event was produced by either the left or the right NAS. This bit is taken into account only if the AM bit is set to '0' since only the NAS module was configured as stereo.
- Bits 7 to 1: frequency channel (FCH). This 7 bits are used for identifying the NAS frequency channel from which the event was produced. Therefore, a maximum of 128 frequency channels can be identified. This value is used for all the submodules. For the NAS module, it indicates the SBPF that produced an output event. For the SOC module, it indicates the frequency channel associated to the neuron that produced the event.
- Bit 0: event polarity (POL). This bit is used for indicating if the output event is positive ('0') or negative ('1'). This bit is taken into account if the AM bit is set to '1' or if the 'AM' bit is set to '0' and the XSO bit is set to '1', that means that either the event was produced by the NAS or by the LSO modules.

For the final analysis, an FPGA resources consumption report of the NAC module was generated for the Artix-7 FPGA chip. A 64-frequency-channel, cascade, stereo NAS was generated using OpenNAS. The input interface was the I2S protocol, and the output interface was the AER protocol. In addition, a SOC model was created. It consisted of a MSO model with a global detection time of 700 μs, 10 frequency channels (channels' addresses from 25 to 34), 16 neurons for each channel, and 5 μs of overlapping between neurons. On the contrary, no LSO module was added for this analysis. Results were collected in Table 4.6.

TABLE 4.6: Hardware resources utilization of the NAC module for the Artix-7 FPGA chip.

Module name	Slice Registers Used / Available	Slice LUTs Used / Available
NAS	20278/94400 (21.48%)	32547/47200 (68.96%)
SOC	7909/94400 (8.38%)	13978/47200 (29.61%)
AER merger	8/94400 (<0.01%)	24/47200 (0.05%)
Total	28195/94400 (29.87%)	46546/47200 (98.61%)

As it was discussed before, the SOC module (containing only the MSO model) requires almost one third of the available FPGA's slice LUTs, while the NAS modules requires more than two thirds of the FPGA's slice LUTs. Therefore, there is no more resources available in this FPGA chip for implementing any other module, as the LSO module. In addition, it is not possible neither to improve the

MSO module by adding either more ITD detection networks or more neurons per ITD detection network in this FPGA.

The project was then generated for the XC7K480T FPGA chip (Kintex-7), obtaining a global slice LUTs utilization of 15.59% and 4.72% of slice registers. Analyzed by module, the NAS needed 10.91% of the available slice LUTs and 3.40% of the slice registers, while the SOC module required 4.68% of the slice LUTs and 1.32% of the slice registers. Consequently, the Kintex-7 family would be needed for implementing more complex models (i.e., with more neurons or frequency channels) or more complete models, including the LSO module.

4.2.6 Analysis and results

Differently to the previous modules that were analyzed and characterized by behavioral simulations, the integration of the SOC module with the NAS was directly analyzed by means of real-time experiments. Among other reasons, one of the main drawbacks is the time needed for completing the simulation, which could be a couple of hours just for one audio file of 0.5 seconds. Taking into account a dataset composed by more than 300 audio files, the simulation process will take too long just for one model configuration, meaning that this option is not viable.

Another disadvantage is the lack of realism when a system is being characterized in simulation. The absence of random noise, introduced by different sources, implies that the model's response is almost perfect. However, that situation does not match with the reality, where the noise plays a key role when working with audio and sound source localization. Therefore, real-time experiments were carried out for characterizing the NAC module by analyzing the module's output in three different aspects: the model configuration, the frequency of the input stimulus, and the distance between the sound source and the microphone pair.

As it was already depicted in Section 4.2.2.5, a virtual room with a pair of microphones was created by using the RIR generator (Habets, 2006). The room's dimensions were again $10 \times 10 \times 4$ meters (x, y, and z). The microphone pair (subcardioid type, oriented as the human ears) was placed at (5, 3, 2). The sound velocity was set to 340 m/s, the reverberation time was set to 0.2 seconds, and the sample frequency was set to 48000 Hz.

For this analysis, multiple pure tones from multiple locations and different distances were generated, thus obtaining a test dataset that was used for the analysis of the model. Three different distances were established: 0.5 meters, 1.0 meters, and 1.5 meters. For each distance, nine positions were defined in a semicircular shape between -90 degrees (completely on the left) to 90 degrees (completely on the right), using as reference the microphone pair, as shown in Fig. 4.23. Then, for each distance and for each position, a set of ten pure tones were

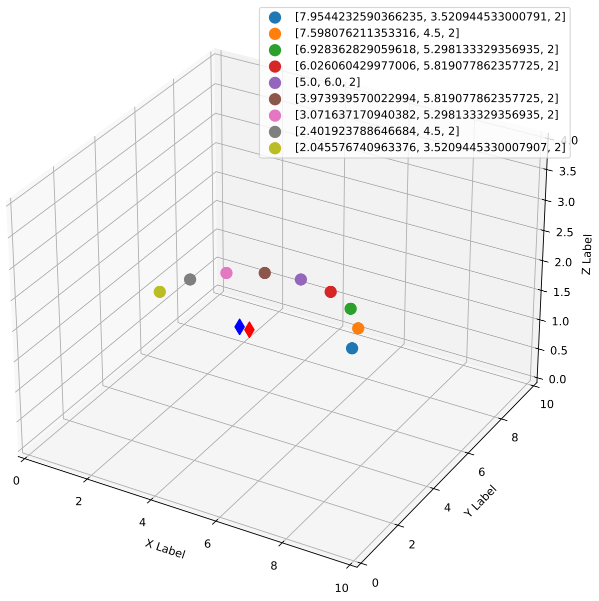


FIGURE 4.23: Virtual room created for testing the NAC. Diamond shape indicates the microphones pair, where blue diamond correspond to the left microphone and red diamond correspond to the right microphone. Therefore, sound sources are placed in front of the microphones pair. The sound source 0 is the one placed completely on the right.

played: 250 Hz, 500 Hz, 750 Hz, 1000 Hz, 1250 Hz, 1500 Hz, 1750 Hz, 2000 Hz, 2250 Hz, and 2500 Hz. The RIR obtained for each pure tone was then convolved and converted to audio files with .wav extension for being later played with a Matlab script. With it, the whole dataset can be played automatically, and the individual response can be saved as an .aedat file for the analysis[8].

Two different NAC configurations were used, where the NAS model was the same and the MSO module within the SOC module was modified. In the first configuration, 10 ITD detection networks, connected to frequency channels from the 25th to the 34th, were used, with 16 coincidence detector neurons for each network. In the second configuration, 4 ITD detection networks, connected to frequency channels from the 25th to the 29th, were used, with 32 coincidence detector neurons for each network. In both cases, the global detection time was set to 700 μs, with an overlapping time of 10 μs.

Two different NAC configurations were used, where the NAS model was

[8]The virtual room generation, as well as the Matlab scripts for this analysis can be found in `https://github.com/dgutierrezATC/nssoc/tree/master/nssoc/Examples/NAS_SSOC/NAS_I2S_64ch_mono_12att_monitor_ztex/realtest`.

the same and the MSO module within the SOC module was modified. In the first configuration (identified as **"model 1"**), 4 ITD detection networks, connected to frequency channels from the 25th to the 29th, were used, with 32 coincidence detector neurons for each network. In the second configuration (identified as **"model 2"**), 10 ITD detection networks, connected to frequency channels from the 25th to the 34th, were used, with 16 coincidence detector neurons for each network. In both cases, the global detection time was set to 700 μs, with an overlapping time of 10 μs. The middle frequencies for each channel, from channel 25 to 24, were: 1366.22, 1222.49, 1093.88, 978.80, 875.82, 783.68, 701.24, 627.46, 561.45, and 502.38, respectively and expressed in Hz.

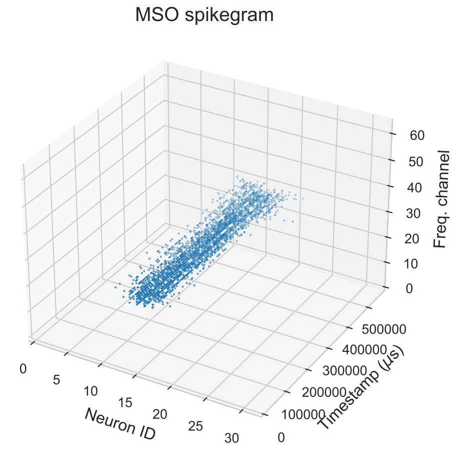
In total, 540 .aedat files were generated. Next subsections will show some representative examples for comparing and analyzing the impact of different models' configurations over the three aspects already mentioned: the configuration of the model, the frequency, and the distance. A full analysis, taking into account all the files, was not carried out due to the high number of samples. In addition, the analysis would depend on the final application, the input stimuli, and the model configuration. For this reason, this qualitative analysis was performed. Nevertheless, an individual analysis for each .aedat file was automatically generated using pyNAVIS tool, and they are available on the GitHub project's repository.

4.2.6.1 How does the model affect?

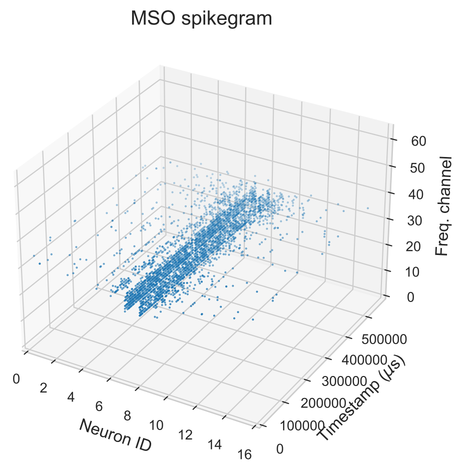
One of the most important decisions before starting a new application is the model's configuration to be used. This decision directly depends on the application's nature, since the parameters of the model should match with the application's requirements. For example, if a high precision is needed, the number of coincidence detector neurons for each ITD detection network should increase, thus decreasing the number of ITD detection networks in order to balance the FPGA resources consumption. This also implies to the model for being more selective in the frequency range from where the ITD wants to be extracted. On the other hand, a MSO model with fewer neurons per network but with more networks would allow covering a wider frequency range, thus being useful for detecting multiple sound sources in different frequencies.

In this case, the same test file was used for both models. It consisted of a pure tone of 750 Hz, played for 0.5 seconds, placed at position 5 over 9 (i.e., in front of the reference with an angle of 0 degrees) with a distance of 0.5 meters. Then, for each result file, three different plots were generated: the MSO spikegram, showing the raw output events of the model, the MSO heatmap, showing the activity of each neuron for each network, and the MSO position estimation, which gives an idea of the accuracy of the model. These six plots are shown in Fig. 4.24.

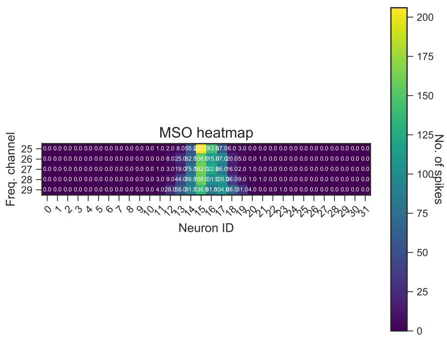
It can be observed in the spikegram that model 1 produces almost the same number of spikes compared to model 2. About the events' distribution, the



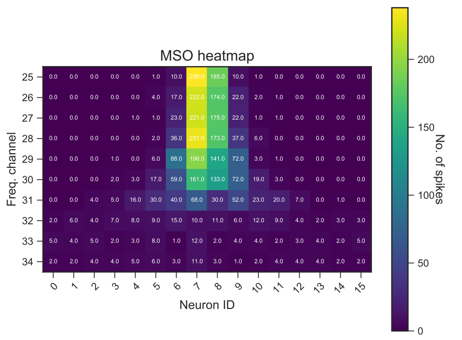
(A) MSO spikegram from model 1.



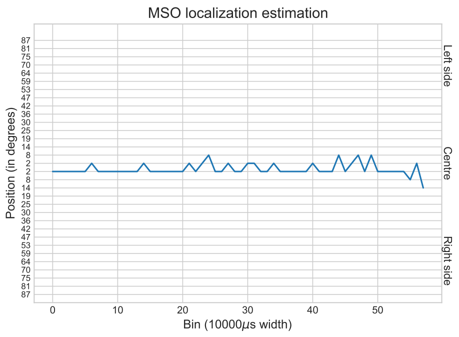
(B) MSO spikegram from model 2.



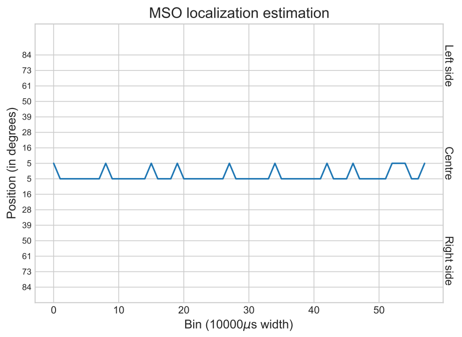
(C) MSO heatmap from model 1.



(D) MSO heatmap from model 2.



(E) MSO localization estimation from model 1.



(F) MSO localization estimation from model 2.

FIGURE 4.24: Comparison between different configurations of the MSO.

output events from model 1 are more clustered around neurons 15 and 16 (which are the middle neurons, corresponding to the position of the sound source), while model 2 produced a more sparse response. Those events which are far from the main cluster could be considered as noise, and it is a direct consequence of the model's configuration. Therefore, it can be confirmed that the model's accuracy directly depends on the number of neurons, as it was expected according to the state-of-the-art.

Furthermore, the heatmaps show almost identical global behavior, presenting both models a cone-like response. The highest activity was found on the peak of the cone, starting to be more sparse while decreasing the middle frequency of the frequency channels. Finally, the sound source position estimation shows that both models are able to determine, with the same accuracy, the real position of the sound source. The oscillations in the estimation were produced because the number of neurons was not an odd number, thus being both positions considered as winners.

Summarizing, the model's parameters have a crucial influence on its response, and they should be chosen according to the final application. The global response of the model would be almost the same, and the final decision would depend on the preliminary results obtained from the first tests carried out by the user. From that point, the model could be finely tuned to match the application's requirements. In our case, model 2 was selected for performing the rest of the tests since it covers more frequency channels and, for this application, the accuracy is not relevant.

4.2.6.2 How does the frequency affect?

As it was mentioned in Section 1.3.1.2.2, the MSO works better for low frequencies (in the range of 300 Hz to 1200 Hz). From that frequency to higher frequencies, the period of the sound becomes equal or lower than the coincidence detection time, thus producing ambiguities in the coincidence detection that can be corrected later by the LSO. This situation could be also improved by decreasing the coincidence detection time in the ITD detection networks associated to higher frequency channels. Frequently, the tuning process of the model's parameters is performed by mean of a training procedure (Liu et al., 2013). However, since the learning process implies the development of the learning algorithm within the system, it was not implemented in this work, and the configuration process was carried out by hand.

To this end, two different situations were then analyzed in order to verify the response of model 2 to this kind of situation where the frequency of the input sound has either low or high frequency. The first analyzed case was the NAC's output response for a pure tone of 500 Hz, placed at position 3 over 9 (30 degrees approximately from the very right position) with a distance of 0.5 meters with respect to the reference. The second analyzed case was the NAC's output response for a pure tone of 1500 Hz, also placed at position 3 over 9 with a distance of 0.5 meters with respect to the reference. The resultant plots are shown in Fig. 4.25.

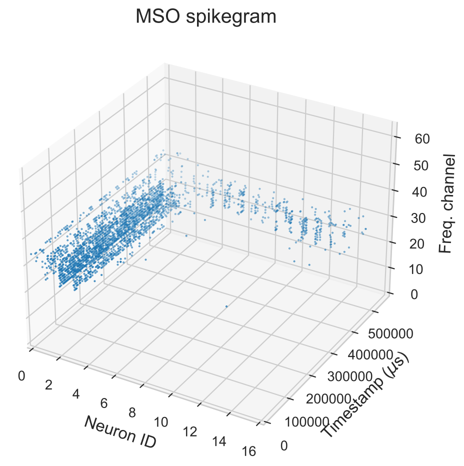
From the spikegram, it can be already observed that the model's response to the high frequency sound (Fig. 4.25b) does not offer any localization information since there is not any main cluster of points. On the contrary, the spikegram of the low frequency sound clearly shows the sound position through the model's response, concentrating all the produced events around neuron 3. Moreover, the heatmaps offer a better view about the behavior of the models. For the 500 Hz pure tone, the system was able to determine that the position of the sound source was more to the right, while for the 1500 Hz pure tone, the system was not able to determine any region of potential localization. In fact, three different high activity points can be identified, meaning that the ambiguity was too high, probably because the distance between the microphones.

Taking a look at the localization estimation plots, it can be seen that the estimation is highly precise for the case showed in Fig. 4.25e. The plot shows that the sound source is placed at 60 degrees to the right with respect to the reference, that matches with the expected result. The final oscillations were produced because the noise at the end of the audio, thus it should not be taken into account. However, if we take a look at the localization estimation for the 1500 Hz sound case, the result is a fully oscillatory behavior without any accurate estimation. This effect could be produced due to the distance between microphones does not match the global coincidence detection time. Theoretically, the MSO's response should be acceptable with sounds up to 1500 Hz, although for this configuration that limit was lower.

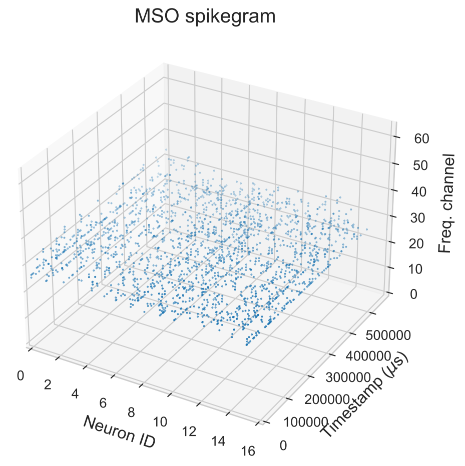
Therefore, it can be said that the frequency of the input sound is a critical aspect to take into account for potential applications, like speaker localization or auditory attention. The maximum accuracy of the system, expressed in degrees, is determined just by the number of coincidence detector neurons, and it can be described as in Equation 4.4.

$$max_{accuracy} = \frac{180}{NUM\_NEURONS} \tag{4.4}$$

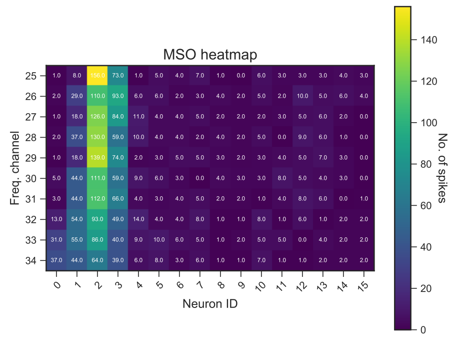
The accuracy is typically measured for the front side with respect to the reference since the back side follows the same principles than the front side but with the difference that the sound's power is lower due to the ear's shape. Another factor that affects on the accuracy is the distance between the reference and the sound source, and it will be analyzed next.



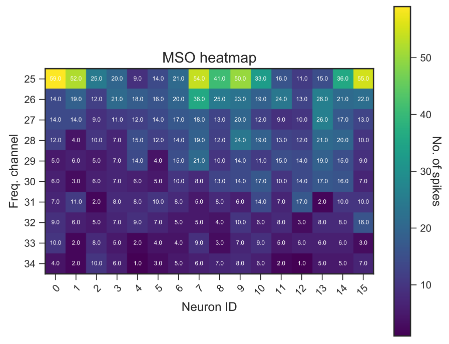
(A) Output spikegram for 500 Hz input sound.



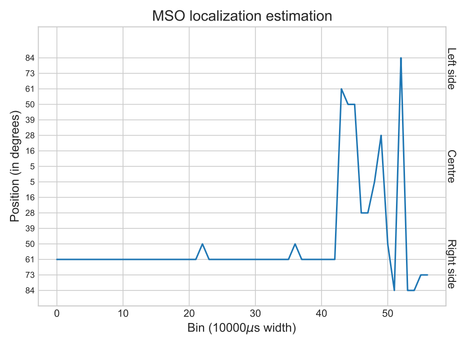
(B) Output spikegram for 1500 Hz input sound.



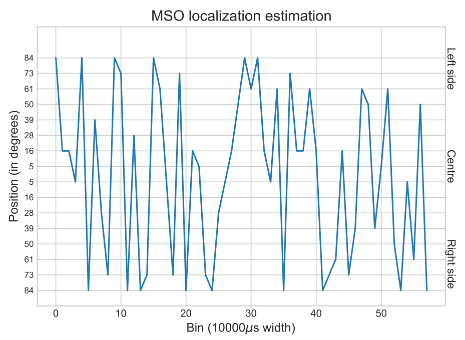
(C) Output heatmap for 500 Hz input sound.



(D) Output heatmap for 1500 Hz input sound.



(E) Localization estimation for 500 Hz input sound.



(F) Localization estimation for 1500 Hz input sound.

FIGURE 4.25: Frequency comparison using model 2.

4.2.6.3 How does the distance affect?

The distance at which a source is producing the sound is important for many reasons. For instance, the sound wave loses some power while it is being propagated through the air, as well as it is expanded in all the directions. Those effects are especially notable in low-frequency sounds, thus coinciding with the frequency range in which the MSO works better. This situation could lead to situations in which the model becomes less accurate due to the ambiguity.

According to (Risoud et al., 2018), the ITD and ILD provide precise localization in the azimuthal plane, with the exception of what is known as the "cone of confusion". For sounds coming from the circumference of this cone, the axis of which is the interauricular line, there are no time or level differences, leading to confusing perceptual coordinates: the subject is unable to tell whether the sound is coming from the front or from behind, above or below, or from anywhere else along the circumference shown in Fig. 4.26. For any sound source with coordinates Δ, α, θ there is a mirror-image position $(\Delta, 180, -\alpha, -\theta)$ with similar ITD and ILD.

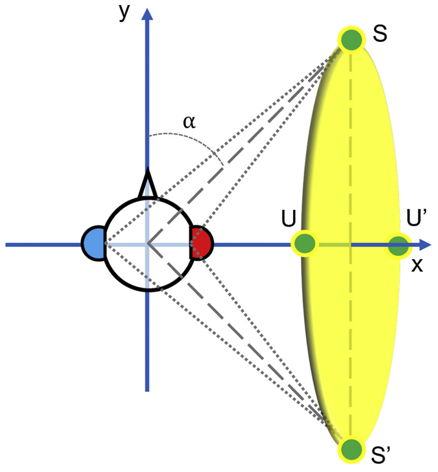


FIGURE 4.26: Concept of cone of confusion. Figure taken from (Risoud et al., 2018).

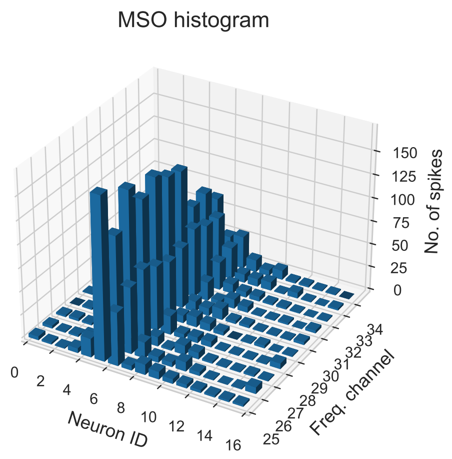
The cone of confusion is critical when analyzing the accuracy of an application and for tuning the localization estimation in robotics applications. There are some metrics that measure the ambiguity or confusion, such as the one presented by Fischer et al. in (Fischer et al., 2020). However, the concept of accuracy could vary depending on the final application, and it should be first defined and then measured. In addition, there are dynamic cues that can reduce ambiguity. According to (Wallach, 1939) and (Risoud et al., 2018), by moving the head (or, in animals, the ears) extra binaural and spectral cues are introduced, thus enhancing localization. By leaning the head (and thus the vertical interaural axis) or turning it, the amplitude and phase of the sound waves reaching either ear are altered, providing dynamic binaural cues. Head movements also provide an accumulation of Head-Related Transfer Functions (HRTFs) and the creation of multiple cone of confusions, refining localization by combining the information and thus constraining probability of a sound source location to a small number of possible sources.

Risoud et al. concludes in (Risoud et al., 2018) that, like for other sensory stimuli, auditory perceptual disambiguation also involves integrating multiple other sensory input, notably visual. Once a sound has been located as coming from an specific area and a specific distance, visual data help fix the position. Moreover, prior knowledge concerning the location of the sound source helps determine its present location. It is noteworthy that images take precedence over sounds, in the "proximity-image effect" described by Gardner in 1968 (Gardner, 1968): if some visual clue is available, the sound source will be located accordingly, even if mistakenly.

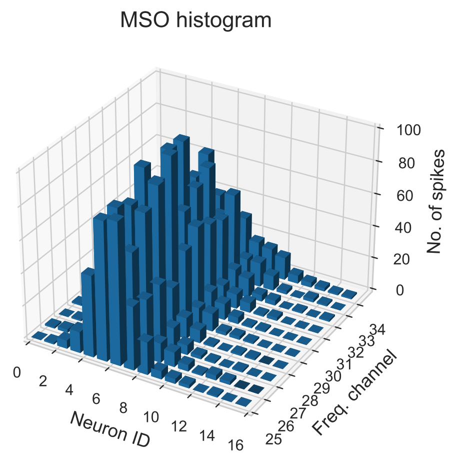
Again, two different situations were analyzed for verifying how the cone of confusion affects to the proposed model. For the first case, the NAC's output response for a pure tone of 500 Hz, placed at the position 4 over 9 (65 degrees approximately from the very right position) with a distance of 0.5 meters with respect to the reference was analyzed. Then, for the second case, the NAC's output response for the same pure tone but with a distance of 3.0 meters with respect to the reference was analyzed. The resultant plots are shown in Fig. 4.27.

Obtained results show the effect of the cone of confusion with the increment of the distance. Comparing both histograms, the activity of the model when the sound source is placed at 0.5 meters is more focused on the winner neuron (neuron 5), while the activity when the sound is placed at 3.0 meters is more distributed over the neuron 5 and its surroundings. In fact, the heatmaps clearly show this behavior. Heatmap associated to a closer distance between the sound source and the reference shows that the winner neuron is the same for all the ITD detection networks, as well as that the difference of the activity with respect to the surrounding neurons is notably higher.

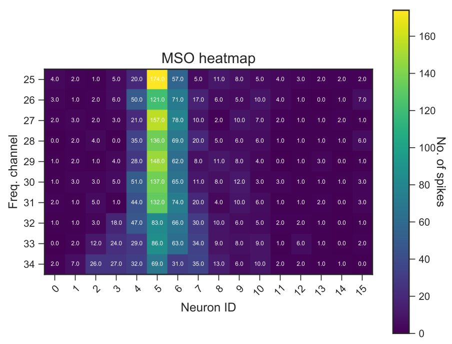
On the contrary, the heatmap associated to the other case presents a sparse activity distribution around neurons 5 and 6, where the range of neurons with notable activity can be established between neurons 3 and 8. Although in this case the ambiguity is not high enough for leading the system to have a clear confusion, it can be seen how the distance directly affects the accuracy of the estimation of the sound source position. This way, it can be observed in Fig. 4.27e that the localization estimation was 100% focused on neuron 5 (62 degrees with respect to the right), without taking into account the last part of the audio. However, for the case where the distance is 3.0 meters, the estimation presents a slightly noisy behavior between four different positions.



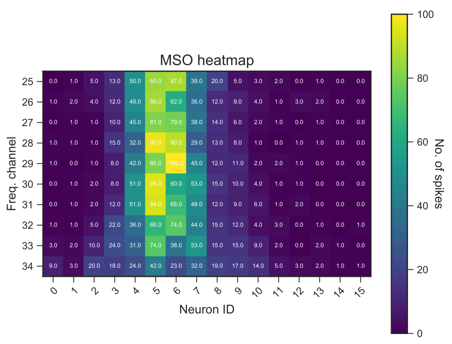
(A) Histogram for a distance of 0.5 meters.



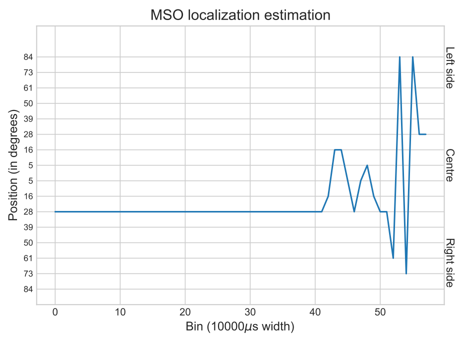
(B) Histogram for a distance of 3.0 meters.



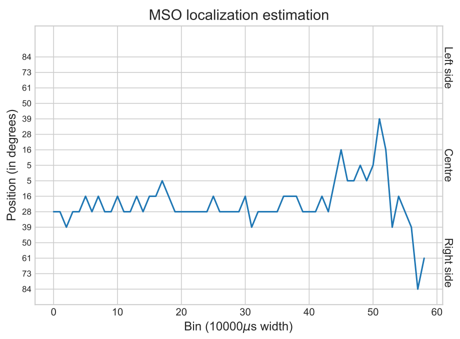
(C) Heatmap for a distance of 0.5 meters.



(D) Heatmap for a distance of 3.0 meters.



(E) Localization estimation for a distance of 0.5 meters.



(F) Localization estimation for a distance of 3.0 meters.

FIGURE 4.27: Effect of the cone of confusion for model 2.

TABLE 4.7: Comparison of the SOC model implementations.

Paper reference	Model/s	Platform	Configuration	Real-time
(Lazzaro and Mead, 1989b)	MSO	ASIC	Manually	No
(Liu et al., 2013)	MSO	Software	Manually	No
(Glackin et al., 2010)	MSO	FPGA	STDP	Yes
(Dávila-Chacón et al., 2018)	MSO, LSO, IC	Software	Bayesian inference	Yes
This work	MSO	FPGA	Manually	Yes

Applied to a robotic platform, this oscillatory behavior could drive a situation in which the oscillation amplitude increases over time with the robot movement, thus finally producing a failure. As it was mentioned before, this problem could be solved by integrating visual information for improving the final estimation and reducing the oscillation. In addition, the localization estimation is not fully performed in the MSO, but it is complemented in the Inferior Colliculus (IC), where a final sound source localization is performed. After that, this auditory information is sent to the Superior Colliculus (SC), where the information is combined with the visual information for taking the final decision. This whole process, although it is complex, would be desired in human-robot interaction applications for improving the user's experience.

4.2.7 Conclusion

In this section, an event-based model of the Superior Olivary Complex (SOC) for performing sound source localization tasks has been presented. This model is part of the development of a completely digital, event-based neuromorphic hearing sense, starting with the NAS and ending with the auditory cortex. By following the same strategy as used in the NAS design, in which spike building blocks (Jimenez-Fernandez et al., 2010) were used, the model was able to successfully extract the ITD from the input stimuli.

In addition, the model was integrated with the NAS, and a set of real-time experiments were carried out for validating the proposed implementation. Table 4.7 compare this work with the works used as reference. Some features, like the implemented models, if it was implemented either in software or hardware, as well as if the model's parameters are tuned by hand or by applying some learning techniques, were qualitatively compared.

Results obtained from the experiments showed that the model implements the features described in Section 1.3.1.2 and in Section 4.1, like the effect of the distance and the effect of the frequency in the MSO model. Further analysis are required for measuring the accuracy of the model, although preliminary results indicates the system is able to determine the position of the sound source very precisely with an error of a few degrees.

The angular resolution of the model could be improved after integrating the output information of the MSO model with the output information of the LSO model. By combining those modules' outputs with a biologically-inspired model of the IC, this sound source localization system could be used for solving complex situations, like auditory attention tasks when multiple sound sources are present (Chou et al., 2019). Finally, the NAC model is suitable to be integrated in many robotic platforms due to its VHDL-based nature, thus allowing it fast integration within any FPGA-based board [9].

4.3 Alternatives to the Jeffress model: The Time Difference Encoder

A recently proposed model which computes temporal dependencies in SNNs is the spiking Elementary Motion Detector (sEMD) (Milde et al., 2018). In that work, the sEMD model consisted of two parts: an event-based vision sensor as input and the TDE as sensory pre-processing unit. The TDE unit translates the time difference between two events into a burst of output spikes. Both the number of output spikes and the duration of the burst produced by the model directly reflect the temporal correlation of two input signals, and it is inversely proportional to the time difference. Milde et al. developed an analog Complementary Metal-Oxide-Semiconductor (CMOS) implementation of the TDE, characterized its performances on silicon and applied it to the encoding of Optical Flow (OF). The TDE model has already been used for processing visual (Schoepe et al., 2019), auditory and olfactory information. Its universal applicability has great potential for inspiring innovative pre-processing for SNNs, especially supporting close-loop neuromorphic systems with low latency requirements.

This wide range of possible applications poses a challenge in terms of time resolution and scalability. Time resolution in analog circuits is constrained by the size of the capacitors. Therefore, for high time constants applications, large capacitors would be needed. Furthermore, mismatch problems and parameter setting difficulties may appear due to the analog nature of the implementation. In this section, a generic, event-based digital implementation of the Time Difference Encoder (TDE) model is presented. Its time resolution is configurable by means of a clock divider, covering a time range from nanoseconds to seconds (Gutierrez-Galan et al., 2021a).

Moreover, the model can be deployed on FPGA-based platforms. This computational platform suits the integration of SNNs very well due to its

[9] A live demo video of this model running in real-time on a robotic platform can be found in `https://youtu.be/v56lpGJEkA4`

highly-parallel, low-latency nature. This TDE implementation facilitates the development of complex and reconfigurable neuromorphic networks receiving input from event-based sensors, such as bio-inspired retinas (Lichtsteiner et al., 2008) and cochleas (Yang et al., 2016; Jimenez-Fernandez et al., 2017; Liu et al., 2010b; Chan et al., 2007). Finally, the TDE's performance was evaluated in simulation by characterizing its response to synthetic input stimuli and also to real world recordings from a Neuromorphic Auditory Sensor (NAS).

Summarizing, the main contributions include the digital TDE model implementation as an alternative of the analog version for event-based, real-time neuromorphic applications with different time constants. To the best of our knowledge, this is the first digital implementation of the TDE model. In addition, the model was fully characterized both in simulation and in an FPGA-based board. A power consumption of 1 mW was measured for a single neuron, and up to 400 units could be deployed in a low-cost FPGA chip. Finally, a proof-of-concept of a sound source lateralization task using the proposed model, where the events were received in real time from a neuromorphic auditory sensor, is presented, providing a new alternative to the state-of-the-art of sound source localization systems. The project is open-source, and the project's repository can be found on GitHub: `https://github.com/dgutierrezATC/TDE_vhdl`.

4.3.1 The Time Difference Encoder model

The TDE model (Milde et al., 2018) translates the temporal difference between two input events into a short burst of output digital pulses. It comprises two inputs: the facilitatory pulse (faci) and the trigger synapse (trig), as well as one spiking output shown in Fig. 4.28a. When an event arrives at the facilitatory input, an exponentially decaying facilitatory variable is generated, called gain. If an event enters the trigger synapse shortly after (i.e., small time difference Δt), as in Fig. 4.28b, an Excitatory Post-Synaptic Current (EPSC) is produced. In this process, the amplitude of the EPSC depends proportionally on the facilitatory variable value i.e., on the gain factor. Therefore, the EPSC's amplitude decreases with increasing time difference (see Equation 4.5).

The trigger synapse projects onto a LIF neuron which integrates the postsynaptic currents in its Membrane Potential (V_{mem}) (see Equation 4.6). Every time V_{mem} reaches the spiking threshold τ_{spike}, a digital output pulse is released. The number of spikes generated is antiproportional to the time difference between the two input spikes (see Fig. 4.28 e). When the time difference between the facilitatory and the trigger pulses is long, the gain value at the time of arrival of the trigger signal is not high enough to generate an EPSC. Thus, no spikes are generated. In case of a negative time difference (an event arrives first at the trigger synapse and then at the facilitatory input, as in Fig. 4.28d, with no output spikes being produced. Therefore, the TDE is direction-selective.

$$I_{e2} = I_{e2} + (w_{e2} \times I_{e1}) \tag{4.5}$$

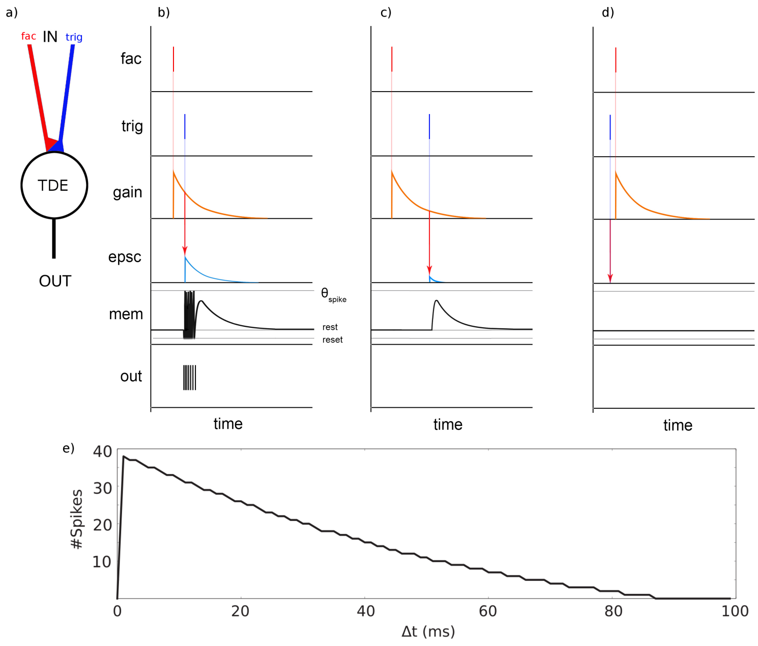


FIGURE 4.28: Theoretical behavior representation of theTDE model based on the model proposed by Milde et al.(Milde et al., 2018). a) TDE schematic with facilitatory (fac) and trigger (trig) input and spiking output. b) Case one: A small positive time difference between facilitatory and trigger spikes leads to a high number of output spikes (out). c) Case two: large positive time difference leads to no output spikes. d) Case three: A negative time difference leads to no output spikes. e) Number of TDE output spikes in dependency of time difference Δt between two input events (gain: gain factor, epsc: exponential postsynaptic current, mem: membrane potential).

$$\frac{dV}{dt} = \frac{1}{C_m} \times \left(I_e - (I_{lk} - I_i) \times \left(1 - e^{\frac{-V}{U_t}} \right) \right) \tag{4.6}$$

As was introduced in section 4.3, both the number of spikes within the burst and the burst duration depend on the time difference between the input pulses. The detection time range of the analog CMOS TDE implementation ranges from 10 nanoseconds up to hundreds of milliseconds, according to (Milde et al., 2018). This range can be tuned by adjusting the LIF neuron's parameters in order to detect the timing differences more precisely in an accurate spectrum, thus obtaining different TDE response profiles. Those profiles are known as tuning curves, which represent the neuron responses against the time difference between the facilitatory and the trigger input pulses.

Furthermore, a nonlinear behavior of the tuning curves was expected to be obtained from the analog CMOS TDE implementation due to the transistors; however, a linear profile was observed. Milde et al. highlighted in (Milde et al., 2018) that the nonlinear response was manifested at the population response level, as well as in the temporal evolution of the ISI distribution within a burst. This feature was taken into account for the digital model design proposed in this work, since it determines the way in which the temporal modules are implemented.

4.3.2 Time Difference Encoder model implementation

The proposed architecture is shown in Fig. 4.29. There are two event-based inputs: the facilitatory input (*"facilitatory"*) and the trigger input (*"trigger"*). In addition, four configuration signals are available to set the model's parameters: the facilitatory weight, called *"detection_time"*, which defines the maximum time difference that the model is able to encode, i.e., the time during which the gain value is non zero; the gain factor that influences the trigger synaptic weight (*"tau"*); the gain factor that influences the spike generation process (*"weight"*); and the decay time factor (*decay*) of the EPSC signal value. As output, there is a single event-based signal (*"spike"*), which is the spike fired by the encoder. Beyond those signals, the system is governed by the system clock signal (*"clock"*).

Both control and event-based signals have 1-bit width. Internal data lines, as well as the *"detection_time"* signal, have n-bit width, with n being a generic parameter of the model denoted by *"NBITS"*. The rest of the data lines have m-bit width, with m being also a generic parameter of the model denoted by *"LOG2NBITS"*, which represents the result of the $\log_2 NBITS$. By default, the *"NBITS"* value is set to 16, thus *"LOG2NBITS"* is set to 4. Data width plays a key role in the model behavior, since it defines the timing resolution and affects the output response due to the implementation details of the spike generator module.

By following the schematic presented by Moritz et al. (Milde et al., 2018), the proposed architecture was divided into three computational blocks: the gain generation, the EPSC generation, and the spike generation, shown in Fig. 4.29 in red, blue, and green, respectively. A phenomenological design strategy was followed to implement the digital TDE model, in order to avoid the computation of differential equations. Therefore, no floating point operations were employed. Instead, integer values were used. This approach was successfully adopted in Frenkel et al. (Frenkel et al., 2017), where linear operations were performed, thus reducing both the hardware cost and the model complexity.

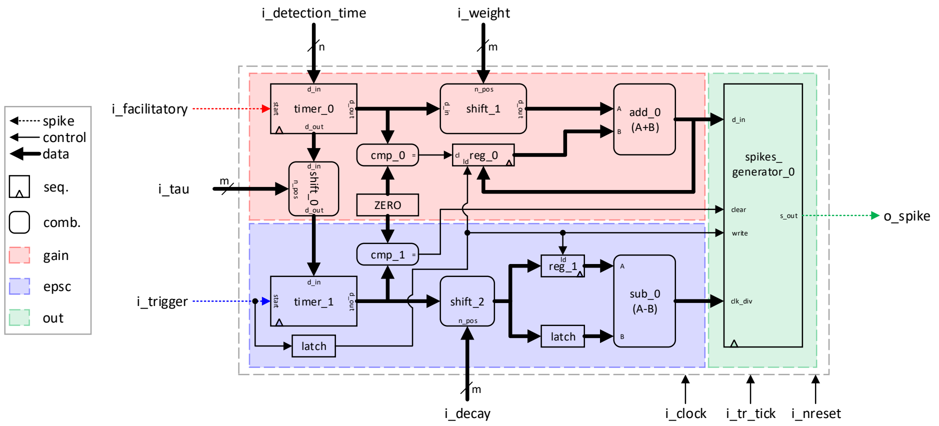


FIGURE 4.29: Detailed block diagram of the TDE digital architecture. It is composed of three main blocks: gain-generator block (red), EPSC-generator block (blue) and spike-generator block (green). Synchronous modules are indicated by squared corner blocks with a small triangle, while asynchronous modules are indicated with rounded corner blocks. The spike arrow is a 1-bit width signal, where events are either received as input or sent as output. Control arrows are also 1-bit width, and they act as flags. Finally, data arrows can be either n-bit or m-bit width, being used for the internal communication between blocks and also for loading configuration values.

4.3.2.1 Gain-generator block

When an event is received at the facilitatory synapse, an exponentially decaying signal is generated (called gain). The decay time constant, as well as the input synaptic weight, determine the maximum time in which the facilitatory synapse current is not zero, i.e., the maximum time difference that the TDE is able to detect. Additionally, if more than one event arrive at the facilitatory input consecutively while the gain is higher than zero, the resulting gain value is the sum of the remaining gain value and the new gain value generated due to the input event. Therefore, a feedback mechanism is needed. In order to prevent the overflow effect, the gain block saturation level is controlled by the *GAIN_GEN_SAT* parameter.

The decaying signal was implemented as a decreasing linear function by means of a countdown timer with pre-load value (represented by the *timer_0* module in Fig. 4.29). The pre-load value establishes the initial configuration of the timer, i.e., the amount of time that the timer is activated. Thus, this temporal window restricts the maximum time difference that the model is able to encode. The input signal *"detection_time"* sets that value, and it can be updated in real-time.

The feedback feature is achieved by internally appending an adder to the timer, where its inputs are the timer's output and the aforementioned *"detection_time"* signal value, and the output is the timer's load value. Therefore, for each rising edge of the time reference signal *"tr_tick"*, the *timer_0* module is updated according to Equation 4.7.

$$timer\_0[k] = \begin{cases} timer\_0[k-1] + d\_t & \text{if } faci == 1 \\ GAIN\_GEN\_SAT & \text{if } satu == 1 \\ timer\_0[k-1] - 1 & \text{if } timer\_0[k-1] > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

Where *timer_0[k]* is the timer's output value at the time reference tick *k*, *k-1* is the previous time reference tick, *d_t* is the unsigned integer constant value defined by the *"detection_time"* signal, *faci* corresponds to the *"facilitatory"* input signal, and *satu* is a flag that is activated when the condition $(timer\_0[k-1] + d\_t) >= GAIN\_GEN\_SAT$ is true.

Two clock domains were used to implement the digital TDE model. The main clock signal, defined in Fig. 4.29 as *"clock"*, governs the control processes of the sequential blocks, as well as the input events detection and the output events generation. Furthermore, a second clock signal, called *"tr_tick"*, is provided as time reference tick in order to allow the model to operate with different time scales, thus achieving an operational time range between nanoseconds and seconds. With this, the model acquires enough flexibility to be used along with a wide set of neuromorphic sensors, which can operate at different time resolutions. No internal clock generator was implemented. Instead, an external configurable clock frequency divider is needed when a TDE module is instantiated. When multiple instances of a TDE unit are present, a shared clock frequency divider can be used rather than a single one per unit, thereby reducing the overall hardware resources consumption and increasing the number of units that can fit into a design.

While in the standard LIF neuron model (Gerstner et al., 2002) each synapse outputs a postsynaptic current that integrates onto the membrane potential, the TDE facilitatory block generates a gain factor that regulates the trigger synapse weight. Therefore, to cover this feature in the proposed architecture, two mechanisms were implemented. Firstly, the *timer_0* output is weighted by the input signal *"tau"* in such a way that the timer value is either right or left shifted by *"tau"* positions. The shift operation was implemented according to the Barrel shifter (Ito, 1989), represented by Equation 4.8.

$$d\_out = \begin{cases} d\_in * 2^{n\_pos} & \text{if } l\_r == 0 \\ d\_in / 2^{n\_pos} & \text{if } l\_r == 1 \end{cases} \tag{4.8}$$

where *d_out* is the output value, *d_in* is the input data, *n_pos* is the number of positions to shift the input data, and *l_r* is for selecting whether the signal has to be shifted either to the left or to the right. Since this is a combinational circuit, the output result is available at the same clock cycle, and thus sequential blocks are not required for the synchronization. The computed value is then fed as input of *timer_1*, which generates the EPSC signal, acting as the trigger synapse weight.

Secondly, *timer_0*'s output value is weighted by the input signal *"weight"* also through a Barrel shifter module. In this case, the result influences the spike generation process in such a way that it controls the number of spikes to be generated and, therefore, the precision of the encoding.

The shifted value is read by the spike generator block when a trigger pulse is detected. Based on the operating principles of the TDE model, detailed in section 4.3.1, it can be deduced that the ratio of the number of spikes to the duration of the whole burst is proportional to the time difference between the facilitatory input and the trigger input. For short time differences, the model will produce many output spikes over a longer time bin, and for long time differences it will produce less spikes but in a shorter time bin.

A register is included to store the last value used as input for the spike generator. Thus, it can be used as feedback value to be added to the gain value, increasing the final gain value and, therefore, increasing the output spike rate. Equation 4.9 describes the gain feedback register, identified in Fig. 4.29 as *reg_0*.

$$reg\_0 = \begin{cases} d\_in & \text{if } trigger == 1 \\ 0 & \text{if } timer\_0[k] == 0 \\ reg\_0 & \text{otherwise.} \end{cases} \tag{4.9}$$

Where *reg_0* is the value stored by the register, *d_in* is the last value loaded on the spike generator block, *trigger* corresponds to the input signal *"trigger"*, and *timer_0[k]* is the timer's output value at the time reference tick *k*. This register is reset to zero when the *timer_0* reaches zero, and it can also be disabled if needed.

Conceptually, it can be affirmed that the gain value influences both the temporal aspect (through the *tau* factor) and the amplitude aspect (through the *weight* factor) of the TDE response. Fig. 4.30 shows a response example of the gain-generator block when both single and multiple facilitatory inputs are provided.

4.3.2.2 EPSC-generator block

Similarly to the gain factor generation, when an incoming event is detected at the trigger's input by the analog implementation from (Milde et al., 2018), an exponentially decaying signal is generated, known as Excitatory Post-Synaptic Current (EPSC). In the field of Neuroscience, the EPSC is defined as the current

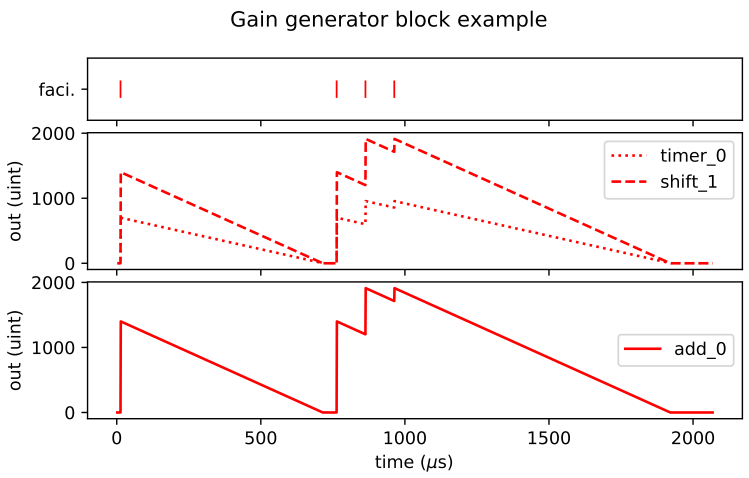


FIGURE 4.30: Gain-generator block output example. First, the block receives a single facilitatory event. Immediately after, *timer_0* is loaded with the *"detection_time"* signal's value. The output value of *timer_0* decreases by one unit for each time reference tick, which was set to microseconds. Since the *"weight"*'s value was set to 1 (meaning that the timer's value is left shifted by one position), the *shift_1*'s output value is twice the value of *timer_0*. Then, the block receives multiple facilitatory events in order to show the accumulative behavior. *add_0* plot shows the value that is used as input for the spike-generator block. In this case, its value matches the *shift_1*'s value, since no trigger event was received.

coming from an artificial synapse that integrates onto a neuron's membrane potential. The amplitude of the Excitatory Post-Synaptic Current (EPSC) is proportional to the gain signal due to the influence of the facilitatory block over the trigger synaptic weight (Milde et al., 2018). Thus, the smaller the arriving time difference (Δt) between the facilitatory and trigger events, the higher the gain factor and, therefore, the higher the amplitude of the trigger synaptic current.

Consequently, if a trigger pulse is detected without any previous facilitatory pulse, no EPSC current is generated, since the gain factor is zero, as is shown in Fig. 4.28c. Nevertheless, if a trigger pulse is detected shortly after a facilitatory pulse (i.e., low Δt), an EPSC current proportional to the gain signal value at that time is generated. The generated EPSC current is high enough to generate spikes when it is integrated onto the membrane potential, as is shown in 4.28b. Equivalently, if a trigger pulse is detected long after the facilitatory pulse (i.e., large Δt), the resulting EPSC current may not be enough to produce output spikes, as in Fig. 4.28c.

Multiple events can arrive to the trigger synapse while the gain factor is higher than zero, thus producing an accumulated EPSC current signal. The resulting signal is the sum of the left over EPSC current value and the left over weighted gain current value. Therefore, a feedback circuit is needed to limit the output current. The feedback value tends to decrease due to the decaying gain factor. However, a high input spike rate may saturate the EPSC current generation. This saturation level is set by the TDE generic parameter *EPSC_GEN_SAT*.

Following the same implementation principle of the gain generator block, the EPSC decaying signal was implemented as a decreasing linear function also by means of a countdown timer with pre-load value, identified as *timer_1* in Fig. 4.29. In this case, we can affirm that the pre-load value is the remaining time to zero of the gain-generator block timer (*timer_0)*; i.e., the gain current signal is zero. In order to maintain the synchronization with the gain generator block, the *timer_1* module is updated at every rising edge of the time reference signal *"tr_tick"* according to Equation 4.10.

$$timer\_1[k] = \begin{cases} timer\_1[k-1] + \left(timer\_0[k-1] \big/ 2^{tau}\right) & \text{if } trigger == 1 \\ EPSC\_GEN\_SAT & \text{if } satu == 1 \\ timer\_1[k-1] - 1 & \text{if } timer\_1[k-1] > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{4.10}$$

Where *timer_1[k]* is the timer's output value at the time reference tick *k*, *k-1* is the previous time reference tick, *tau* is a factor to weight *timer_0*'s output value, *timer_0[k-1]* is *timer_0*'s output value at the previous time reference tick, *trigger* corresponds to the input signal *"trigger"*, and *satu* is a flag that is activated when the condition $(timer\_1[k-1] + (timer\_0[k-1] * 2^{tau})) >= EPSC\_GEN\_SAT$ is true.

As was previously mentioned, the trigger timer determines the output spike burst duration as the same way the EPSC synaptic current decay in (Milde et al., 2018) is set by a voltage parameter. This synaptic current is injected into the neuron that integrates the current until it generates a spike as soon as the membrane potential rises above its threshold. Therefore, the neuron is able to produce spikes while the EPSC signal is higher than zero. In addition, the number of generated spikes is directly proportional to the EPSC signal duration, and thus inversely proportional to the time difference between the facilitatory and trigger input spikes.

In the proposed design, the value that is loaded in *timer_1* (i.e., in the EPSC generator) is called the remaining time to zero. This value influences the spike-generator block in two similar aspects. Firstly, the spike generator block is active, i.e., producing spikes while the EPSC value is higher than zero. Thus, it acts as an enable signal. Moreover, the remaining time to zero is used to handle the temporal evolution of the spike-generation process in order to mimic the ISI increment of the original analog TDE model (Milde et al., 2018) by exploiting a feature of the Exhaustive Synthetic Spikes Generator models proposed in (Linares-Barranco et al., 2006) and implemented in (Jimenez-Fernandez et al., 2010) and (Gomez-Rodriguez et al., 2005).

The temporal evolution can be adjusted by the factor *"decay"*, which weights the *timer_1*'s output also by means of a Barrel shifter module identified in Fig. 4.29 as *shift_2*. The timer's value decreases by one unit for each *"tr_tick"* rising edge. With this factor, we can scale the decreasing speed, allowing us to obtain a different range of values, although preserving the time bin, i.e., the activation time of the spike generator module. Further details about the effect of this parameter are discussed in the next section.

Due to the implementation details of the spike-generator block, the generated EPSC signal needs to be inverted, thus obtaining an incremental signal instead of a decreasing signal. This transformation can be achieved by storing the reference value and periodically subtracting the original value every time it is updated. In this case, the reference value corresponds to the pre-load value of *timer_1* when a pulse is detected at the *"trigger"* input, and the original value is the *timer_1*'s output value. A generic register, denoted by *reg_1* in Fig. 4.29, was added to the proposed architecture, and its behavior is described by Equation 4.11.

$$reg\_1 = \begin{cases} timer\_1[k] * 2^{decay} & \text{if } trigger == 1 \\ reg\_1 & \text{otherwise.} \end{cases} \tag{4.11}$$

Where *reg_1* is the value stored by the register, *decay* is the factor that weights the *timer_1* value, and *trigger* corresponds to the *"trigger"* input signal.

Note that the *"trigger"* signal is latched to let the timer load the pre-load value and to output the correct value, which takes one clock cycle of the main clock signal (*"clock"*). This latched *"trigger"* signal is shared by *reg_1*, *reg_0*, and the *spike_generator_0* modules to keep the synchronization and to operate with the precise values.

The output of the subtractor module, whose output ranges from zero to *reg_1*'s output value, is then used as input of the spike-generator module, which generates spikes according to both the *add_0*'s output value and the *sub_0*'s output value, i.e., the gain factor value and the EPSC factor value, respectively. Fig. 4.31 shows a response example of the EPSC-generator block when both single and multiple trigger inputs are provided after the arrival of a single facilitatory event.

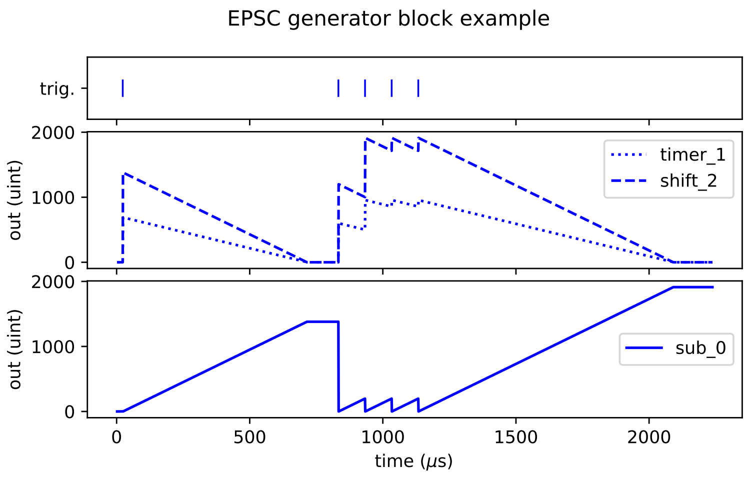


FIGURE 4.31: EPSC-generator block output example. First, the model is stimulated with a single facilitatory event before the first trigger event. Then, the current value of *shift_1* is loaded in *timer_1*. The accumulative effect is also shown when multiple triggers are received. The *sub_0* module generates an increasing signal, which is used as the clock divider value for the spike generator block.

4.3.2.3 Spike-generator block

In the presence of an input facilitatory spike and an input trigger spike, a burst of output spikes is produced by the spike generator block. As detailed in subsections 4.3.2.1 and 4.3.2.2, both the amplitude and the duration of the burst depend on the generated gain factor and the generated EPSC factor respectively. In contrast to the LIF neuron, which integrates the pre-synaptic current into the membrane and produces a spike if the membrane potential reaches a threshold, an event-based unsigned integer-to-spikes converter was implemented based on the model implemented by Jimenez-Fernandez et al. (Jimenez-Fernandez et al., 2010).

This converter, called Exhaustive Unsigned Synthetic Spikes Generator (EU-SSG), takes an unsigned integer value as input and produces a burst of spikes where both the number of output spikes and their distribution over time are proportional to the input value. Similarly to the synaptic current integration process, which polarizes the membrane, producing a membrane potential, the EU-SSG implements an up counter that increases its value every time a pulse is detected, shown in Fig. 4.32 as *up_counter*. Then, the output of the counter is processed by the *Exhaustive Synthetic Method (ESM)* block, which determines the integration method. Finally, analogous to the comparison between the firing threshold and the membrane potential, the counter's output value and the input data are compared, and a spike is fired only when both values are equal.

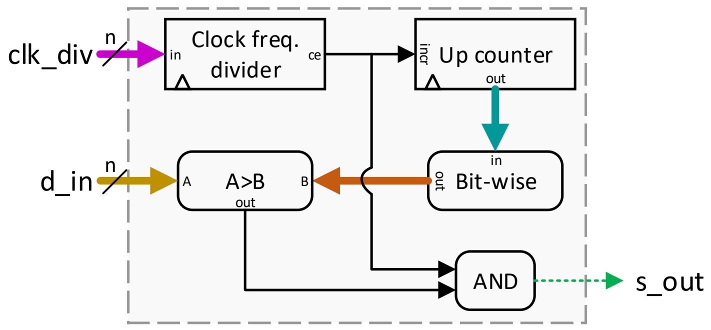


FIGURE 4.32: Exhaustive Unsigned Synthetic Spikes Generator (EU-SSG) block diagram. A complete description of both the implementation and the behavior of this module is presented in (Jimenez-Fernandez et al., 2010).

To generate the pulse according to the desired output firing rate (expressed in spikes per second), a clock frequency divider module is used, identified as *clk_freq_div* in Fig. 4.32. It has two input signals: the system clock *"clock"* signal, and the *"clk_div"* signal, which is the clock frequency division factor; and one output signal, i.e., the clock enable *"ce"* signal, which generates a pulse when corresponding.

By combining both the input value and the clock frequency divider value of the spike generator block properly, the desired behavioral output response of the original TDE model can be achieved. Analytically, the EU-SSG's output firing rate for a given input value was obtained following Equation 4.12.

$$f(d\_in) = \frac{F_{clock} * d\_in}{2^n(clk\_div + 1)} \tag{4.12}$$

Where f is the spike generator's output firing rate (expressed in spikes per second), d_in is the input value to be converted to spikes, the constant F_{clock} is the system clock frequency (in Hz), n is data width, and clk_div is the internal clock frequency divider value.

Although the Equation 4.12 is almost identical to the one defined in (Jimenez-Fernandez et al., 2010), there is a difference in the component 2^n, due to the fact that in the original model the sign is taken into account (2^{n-1} is used instead), while in Equation 4.12 the unsigned version is used. However, the number of bits selected to represent the input value does not affect the maximum firing rate achievable by the EU-SSG block. d_in is the maximum value that can be represented with n bits, which is 2^n. In that case, if d_in is replaced in Equation 4.12, the maximum output firing rate is expressed as in Equation 4.13.

$$f_{max} = \frac{F_{clock}}{(clk\_div + 1)} \tag{4.13}$$

Where f_{max} is the spike generator's output maximum firing rate, the constant F_{clock} is the system clock frequency (in Hz), and *clk_div* is the internal clock frequency divider value. Therefore, the output firing rate only depends on both the system clock frequency and the clock frequency divider value. Since the maximum generating frequency is independent of the number of bits used to represent the values, this parameter can be set up according to the desired time resolution, thus benefiting the resource saving and allowing the implementation of a larger number of TDE units.

Two particular differences can be highlighted from this spike-generator block compared to the LIF neuron model. Firstly, the EU-SSG block does not implement any refractory period. Therefore, the spikes can be produced through consecutive clock cycles. Instead, the clock frequency divider value needs to be adjusted in order to achieve the desired output spikes distribution. Secondly, the spike generator does not stop producing spikes. If the input value is higher than zero, the spike generator continues generating spikes. To stop the spike generation process, a clear input signal *"clear"* was added to the EU-SSG block, which resets its internal registers to zero. This control line is activated by the comparator *cmp_1* when the EPSC's timer output value reaches zero, meaning that the EPSC synaptic current is zero and, therefore, there is no current to integrate.

According to the architecture shown in Fig. 4.29, the EU-SSG input data (*"d_in"*) corresponds to the TDE gain value. This signal depends on the current value of *timer_0*, which determines the detection time. The multiplication factor also controls the number of spikes to generate, as well as the last input value loaded into the generator. Thus, the gain block's output signal, and therefore the *"d_in"* signal, can be defined as in Equation 4.14.

$$d\_in = reg\_0 + (2^{weight} * timer\_0) \tag{4.14}$$

Where *d_in* is the spike generator input value, *reg_0* is the value stored in the register defined by Equation 4.9, *weight* represents the *timer_0* factor, and *timer_0* is the timer value defined by Equation 4.7.

In the same way, the EU-SSG clock divider value (*"clk_div"*) corresponds to the TDE EPSC value. It depends on the current value of *timer_1*, which in turn depends on *timer_0* and *shift_0*, according to Equation 4.10. In addition, the *timer_1*'s value is weighted to control the output firing rate and the ISI variation. Therefore, the EPSC block's output value can be defined as in Equation 4.15.

$$clk\_div = reg\_1 - (2^{decay} * timer\_1) \tag{4.15}$$

Where *clk_div* is the spike generator input value, *reg_1* is the value stored in the register defined by Equation 4.11, *decay* represents the *timer_1* factor, and *timer_1* is the timer value defined by Equation 4.10.

A behavioral example of the proposed model is shown in Fig. 4.33. The first row shows the input events differentiated by colors and following the color code used in Fig. 4.28: red color for the facilitatory event and blue for the trigger event. The second and third rows represent the evolution of the gain and EPSC timers over time, respectively. The fourth row illustrates the *"d_in"* signal value loaded into the spike generator block, and the fifth row represents the clock divider value of that block. Finally, the sixth row shows the output spikes produced in the presence of the shown input stimuli.

When an event is detected at the facilitatory input, the gain timer is initialized according to its detection time value. For each time reference tick, the gain timer decreases its value. As soon as an event is detected, the current value of the gain timer is loaded into the EPSC timer. Concurrently, that value is weighted and loaded into the spike generator as input data. Therefore, the spikes begin to be generated.

At this moment, the clock divider value of the spike generator is set to zero, thus producing the spikes at the maximal firing rate. For each time reference tick, the EPSC timer decreases its value and the clock divider value of the spike-generator block is updated to a higher value. This causes a decrement in the output firing rate and, consequently, an increment of the ISI between two consecutive output events.

When another spike is detected at the trigger input, the current value of the facilitatory timer is added to the current value of the trigger timer. The input data of the spike generator block is updated, and the clock divider value is set to zero. This leads to a reset of the internal counter of the spike-generator block, which leads to an update of the ISI value according to the new input value. The TDE produces spikes until the EPSC timer reaches zero, when the internal stop signal is enabled. A more exhaustive behavioural analysis of the TDE response to both simple and complex stimuli is presented in the following sections.

4.3.3 Analysis and results

Three test scenarios were considered in order to validate and analyse the proposed model. Firstly, a behavioral simulation was performed for the most common input stimulus combinations. The results were cross-validated with the results presented in the reference work (Milde et al., 2018). Once validated, the model was analyzed and characterized by carrying out different experiments

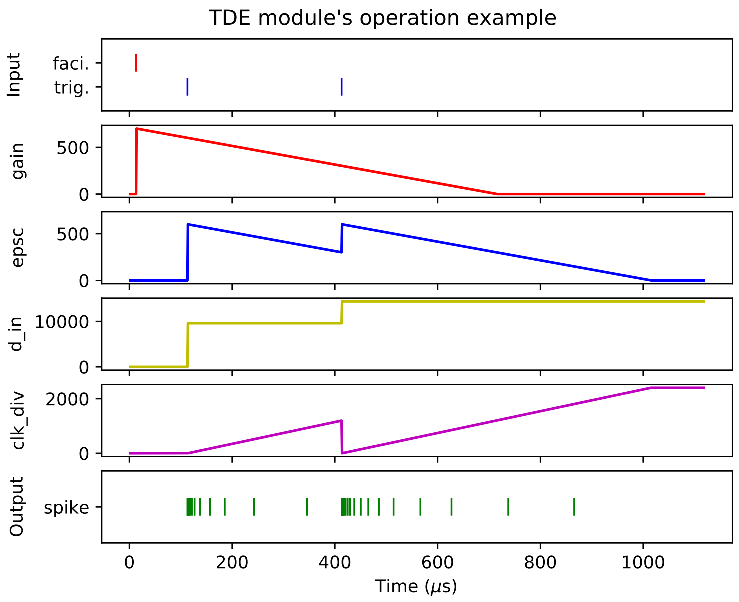


FIGURE 4.33: An operation example of the proposed model. The detection time was set to 700 μs; the tau value was set to 0; the weight value was set to 4; and the decay value was set to 2. The main clock was set to 50 MHz, the time reference tick was set to 1 MHz, and the data width was set to 16. The Δt value between the facilitatory event and the first trigger event is 200 μs, while the Δt value between the facilitatory event and the second trigger event is 400 μs.

where the ISI distribution and the number of output spikes were measured. Secondly, a single TDE unit was synthesized for an FPGA platform. The output spikes obtained from the FPGA were measured using an oscilloscope and recorded using a computer. A quantitative comparison was carried out between the simulation and the deployed version of the TDE unit. Thirdly, a proof-of-concept of a sound source lateralization system was designed and tested using a population of TDE units.

4.3.3.1 Simulation

Since the TDE model has two inputs, many different input event combinations can occur. It is important to study each of these scenarios, since they will directly affect the behavior of the model and its response. With the aim of verifying whether the behavior of the proposed model matches the expected output, twelve cases were simulated. For this experiment, a single TDE unit was instantiated, with 100 μs as detection time, a tau value of 0, a gain value of 5, a decay value of 1, and a value of 256 for both facilitatory and trigger saturation. In addition, the time resolution was set to microseconds. Fig. 4.34 presents the response of the TDE unit when being excited by twelve different sequences of input events.

Cases A and B depict simple examples where either a single facilitatory or trigger event is received by the TDE unit. No events were produced at the output. However, while the gain signal started being generated in case A, as a response to the facilitatory event, the trigger signal remained at zero in case B, since no facilitatory event was received before. Indeed, this effect can also be seen in case C, where the trigger event is received just a few microseconds before the facilitatory event. The same response of the model is obtained when both events arrive at the TDE unit at the same time, as shown in D.

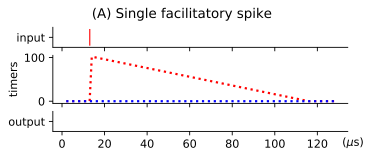
When a facilitatory event is presented at the TDE's input before the trigger event, the TDE's response is inversely proportional to the time difference (also called Δt) between them. Cases E to G show the output events generated by the TDE unit for short, medium and long Δt values, respectively. As can be seen, the number of output events decreases with higher Δt values, while the ISI increases, matching the expected behavior. When Δt is higher than the detection time (case H), the resulting response is the same as having case A first and then case B, meaning that no events are generated at the output.

The proposed model was also simulated and evaluated in the presence of more realistic input patterns. In a real world application, the input events are not received one by one. Instead, a continuous rate of events can be injected to the input. Cases I and J show the TDE model response when multiple facilitatory or trigger events are received at the input for a single opposite event.

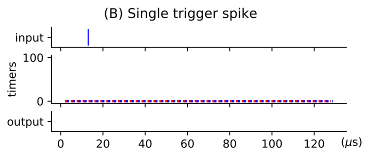
On the one hand, case I shows how the EPSC signal is incremented by a value proportional to the remaining time to zero of the gain signal with the arrival of the second trigger event, following Equation 4.10. The first burst produced contains more events with lower ISI, whereas the opposite happens in the second burst. On the other hand, case J shows how the gain signal is incremented by the detection time value when the second facilitatory event arrives, according to Equation 4.7. This alters the response of the simple case shown in F, generating a burst with a higher number of events.

When many consecutive facilitatory-trigger pairs with Δt lower than the detection time parameter are received, the accumulated EPSC/gain signal values increase. Thus, after some time, the signals saturate according to the saturation parameter. At the saturation level, the TDE output firing rate is considered maximal, and its behavior can be estimated by using Equation 4.13. We can affirm that the saturation parameter limits the TDE response, and its value would have to be set depending on the application and the related statistics of the input stimuli.

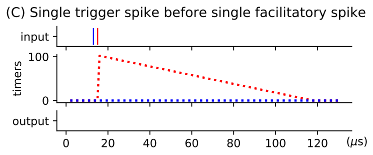
In this regard, the behavioral validation of the proposed TDE model shown in Fig. 4.34 has been proved to act in accordance with the reference model (Milde



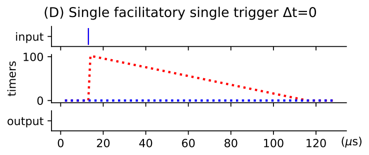
(A)



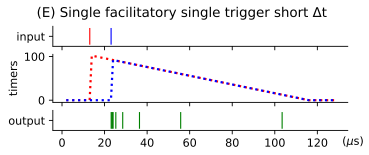
(B)



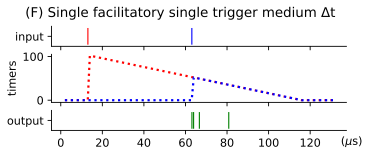
(C)



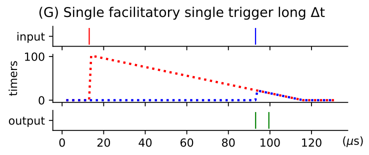
(D)



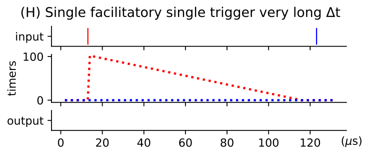
(E)



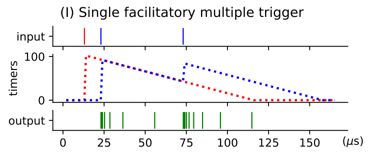
(F)



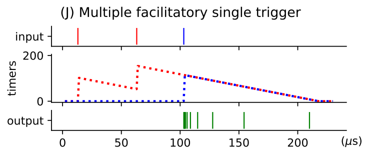
(G)



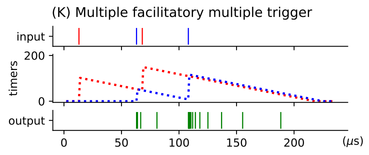
(H)



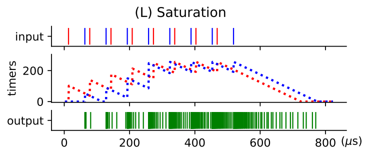
(I)



(J)



(K)



(L)

FIGURE 4.34: RTL simulation for twelve basic cases of the TDE model with a time resolution of microseconds. Red, blue, and green are related to facilitatory, trigger, and spike generator, respectively.

et al., 2018) in terms of both performance and requirements. Moreover, cases which were not evaluated in the original model were also reported in order to fully characterize the proposed model.

After the behavioral validation, a more precise timing analysis of the TDE response was carried out. In (Milde et al., 2018), this study was performed by investigating the ISI distribution within a burst for six different Δt values. The ISI was calculated as $ISI_n = t_n - t_{n-1}$, where t_n is the timestamp of the n_{th} event. The authors reported that the obtained results matched the expected nonlinear response in the temporal evolution of the ISI within a burst.

The same test was done in order to verify that the nonlinear ISI variation feature was also achieved by the proposed model. Fourteen facilitatory-trigger pairs of events with different Δt values were used as input stimuli. Two TDE units were configured to work at different time scales by setting the time reference tick to microseconds and milliseconds, respectively. Fig. 4.35 presents the results obtained from simulations with a time reference tick in the scale of microseconds, instantiating a TDE unit with 700 μs as detection time, 0 as tau, 4 as weight, and 2 as decay. Similarly, Fig. 4.36 presents the results obtained from the simulations, although setting the time reference tick to milliseconds, instantiating a TDE unit with 70 ms as detection time, 0 as tau, 0 as weight, and 3 as decay.

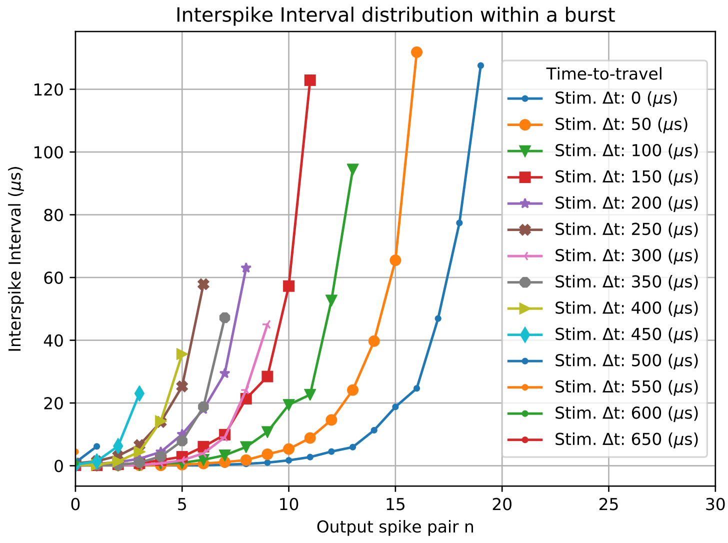


FIGURE 4.35: TDE ISI response for a facilitatory-trigger pair with different Δt values for a microseconds resolution configuration. Note that the smallest Δt value used was not zero (no output events would be produced) but 20 ns (one clock cycle).

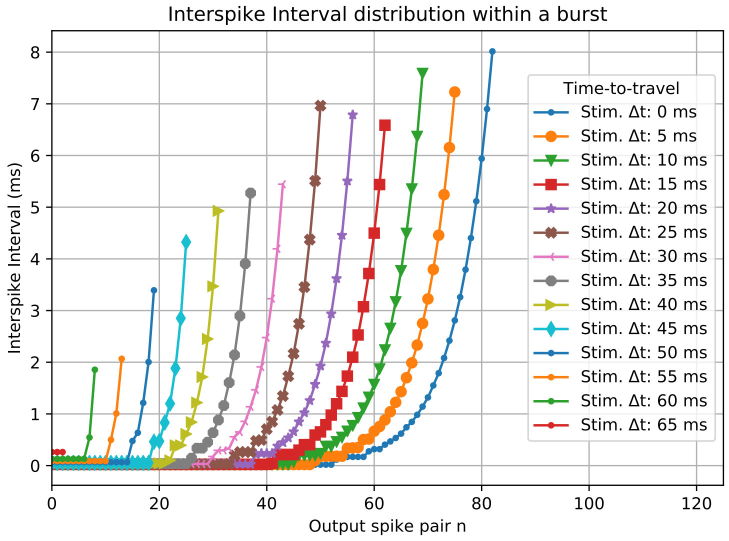


FIGURE 4.36: TDE ISI response for a facilitatory-trigger pair with different Δt for a milliseconds resolution configuration with a detection time of 70 ms.

Similar to the analog CMOS implementation, nonlinear profiles can be clearly observed in all the cases shown in both Fig. 4.35 and 4.36. These profiles were obtained by using exclusively linear operations and circuits, thus avoiding explicit circuitry for generating exponential behaviors. This feature allows reducing the needed resources and therefore to increase the total number of TDE units that can be instantiated into an FPGA or ASIC. When the Δt value is almost equal to the configured detection time (e.g., above 550 μs for microseconds and 55 ms for milliseconds), the produced output events are not enough to represent the characteristic curve that cases with lower Δt presented. On the other hand, the first output event pairs seem to have the same ISI value for most Δt values (specially for lower Δt values). These ISI values cannot be correctly appreciated in the plot, since, according to the global clock, the precision of the minimum time difference is in the order of nanoseconds, and the Y-axis of the plot is represented in milliseconds.

The time scale set by the time reference tick affects not only the ISI curves, which has a better and more regular distribution for the millisecond time reference, but also the number of output events produced. This effect is caused by the combined use of both clock domains in the spike generator module, where the time reference clock is used to manage the inputs and the global clock is used to produce the events.

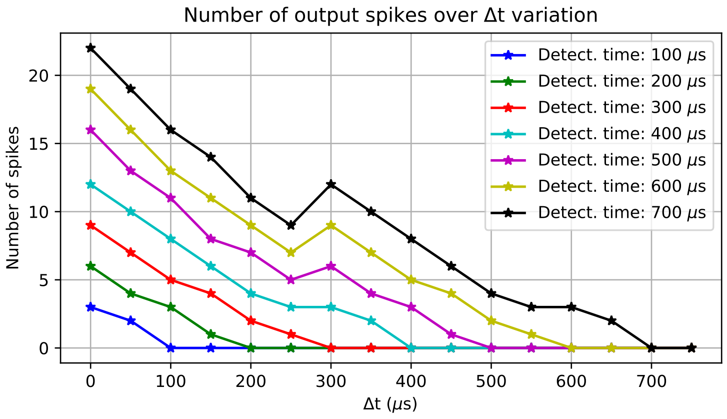


FIGURE 4.37: Number of TDE output spikes over Δt variation for microsecond time reference tick.

Another simulation was carried out in order to prove the variation in the number of output events generated by the TDE unit using both different time references and detection times. The results are depicted in Fig. 4.37 and Fig. 4.38 for a microsecond and a millisecond time reference, respectively.

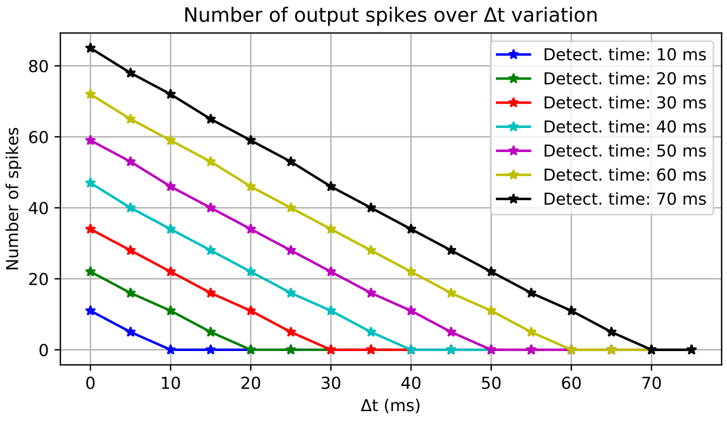


FIGURE 4.38: Number of TDE output spikes over Δt variation for millisecond time reference tick.

Note that the peak located at $\Delta t = 300\mu s$ (shown in Fig. 4.37) is caused by the implementation of the spike generator module, since the conversion from an integer value to a spike stream has an intrinsic error. This error is maximal at that time in this particular example, and it is deeply analyzed in (Gomez-Rodriguez et al., 2005). Due to the timing resolution used in Fig. 4.38, even if the error exists, the peak cannot be appreciated.

The characteristic curve that relates the number of output events produced by the TDE with respect to the Δt value between its facilitatory and trigger inputs is known as the tuning curve. By varying the parameters' values of the TDE unit, its tuning curve can be adjusted. Therefore, it is possible to have a set of TDE units with different tuning curves.

This feature allows configuring a TDE population with different tuning curves responding to different input patterns or using the population response as a global response. Unlike the tuning curve test carried out in (Milde et al., 2018), in which all the neurons shared the same parameters, we conducted a similar test with different TDEs configurations.

Table 4.8 summarizes the values used for each TDE unit within the population created for this test. The population size is four units. All the units share the saturation value, set to 3000 for both the gain and EPSC signals, as well as the *tau* value, which was set to zero.

TABLE 4.8: Parameters of the TDE population

Neuron ID	weight (μs/ms)	decay (μs/ms)	detection time (μs/ms)
TDE 0	9 / 4	2 / 1	100 / 10
TDE 1	6 / 2	1 / 2	300 / 30
TDE 2	5 / 1	2 / 3	500 / 50
TDE 3	3 / 0	2 / 12	700 / 60

The time difference of the two input events was varied from 20 nanoseconds to 750 microseconds for the microsecond time reference, with a 20 microseconds step size. Similarly, the relative timing was varied from 20 nanoseconds to 75 milliseconds for the millisecond time reference, with a 5 milliseconds step size. The results obtained from the simulations are depicted in Fig. 4.39 and Fig. 4.40.

Both figures show the effect of the different tuning parameters in the output response of the TDE units. The TDE0's tuning curve presents a noticeable slope, meaning that it has a high output firing rate in a short time bin. On the contrary, the TDE1's tuning curve has an almost flat slope, which means it produces less spikes but in a longer time period.

The tuning curves in Fig. 4.37 and Fig. 4.38 look practically linear. These tuning curves show a comparable behaviour to the analog model, which is considered non-linear for large temporal differences and linear for small time differences by Milde et al. (Milde et al., 2018).

The global behavior of the population, calculated as the sum of the output events for each TDE unit for each Δt value, fits an exponential curve. For the

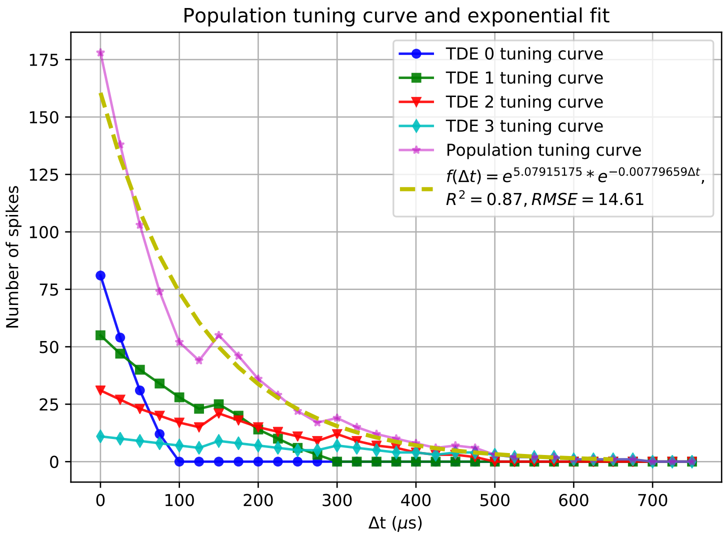


FIGURE 4.39: Individual TDE tuning curves and population tuning curve for microseconds time reference tick.

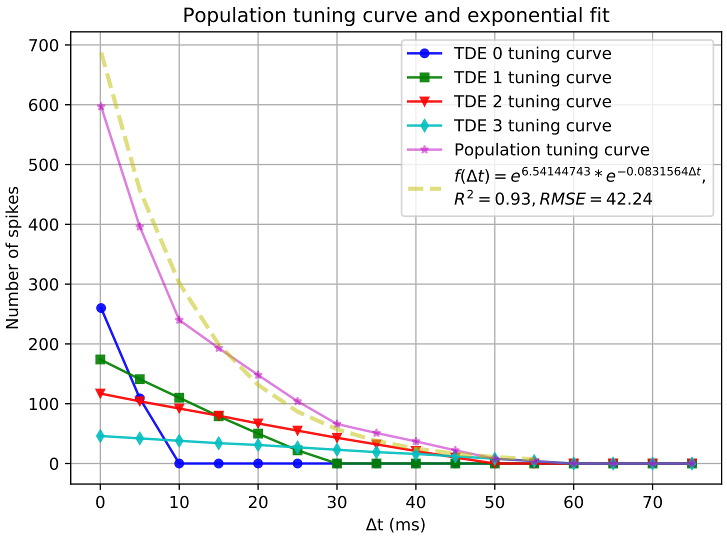


FIGURE 4.40: Individual TDE tuning curves and population tuning curve for microseconds time reference tick.

microseconds time reference, the obtained exponential fitting curve had a $R^2 = 0.87$ with a $RMSE = 14.61$. For the milliseconds case, the R^2 value was 0.93 with a $RMSE = 42.24$.

The spike generator intrinsic error directly effects the exponential approximation, although the fitting curve can be considered acceptable taking into account that a nonlinear profile was obtained by using exclusively linear modules.

4.3.3.2 FPGA implementation

After simulating the proposed model, analyzing and validating its behavior, a TDE population was deployed into a FPGA-based device in order to verify the results obtained in the simulation using a hardware platform. Fig. 4.41 depicts the setup used for this test. The upper part describes in detail the implemented design deployed into the FPGA. Two timers with periodic interruptions were used to generate both the facilitatory and trigger events. The time reference tick was set to microseconds and the Δt value was fixed to 100 μs, having a wait time of 1 second between two consecutive stimulus generations. The population size was set to four in order to maintain the same architecture as in the simulation. Therefore, a ROM module was added for storing the population parameters, which were the same as those presented in Table 4.8 for the microseconds case. Finally, an events monitor was connected to the population output to collect the events and to send them to the computer by using an AER protocol.

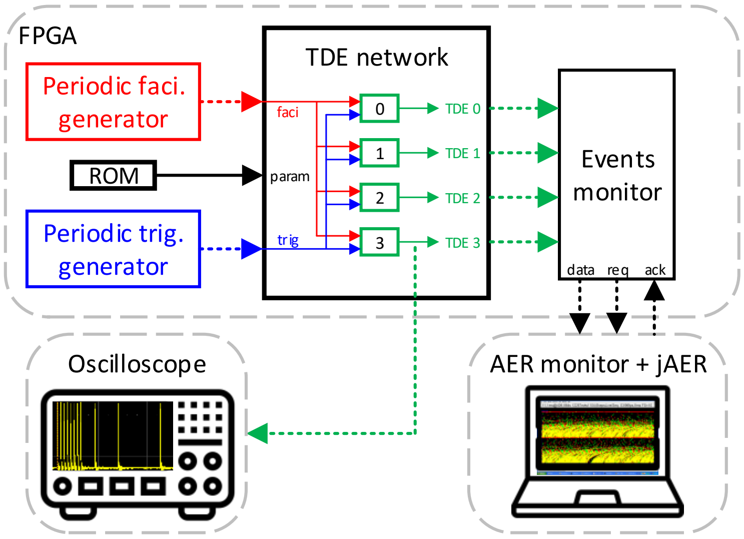


FIGURE 4.41: Block diagram of the setup for real-time measurements acquisition.

The lower part of Fig. 4.41 describes the two approaches used to measure the population response directly from the hardware. On the one hand, an oscilloscope was used to both measure and visualize the output events from the TDE3. Fig. 4.42 shows a screenshot with the captured events, where the increment of the ISI over time can be appreciated. On the other hand, a computer running jAER (Delbruck, 2007) was used to visualize and save the population output events in real time.

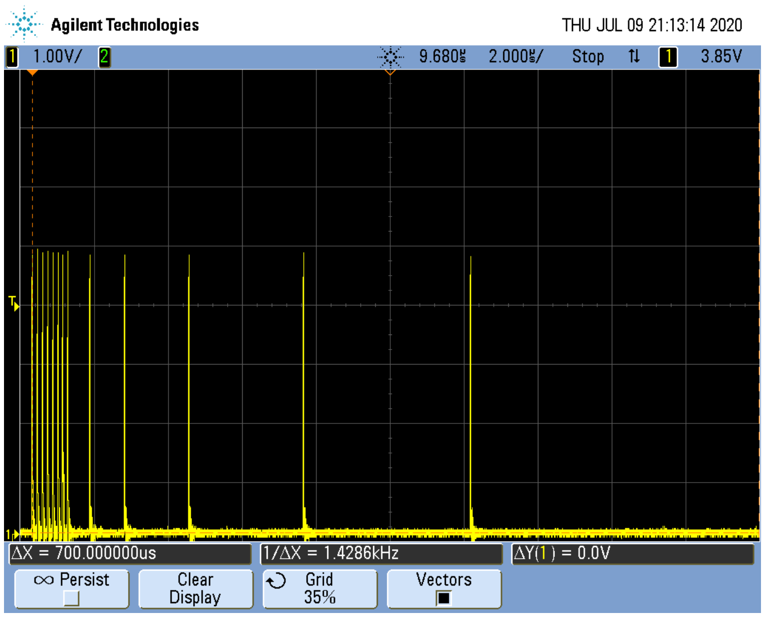


FIGURE 4.42: Output spikes captured by using an oscilloscope.

In addition to the behavioral simulation, a post-synthesis simulation and a post-implementation simulation were performed. The behavioural simulation was used as a reference and compared to the simulation results, as well as the measurements from the oscilloscope and the events collected by the events monitor. The output of TDE3 was used to compute the Pearson correlation value (Benesty et al., 2009) for a quantitative comparison.

The results are plotted in Fig. 4.43, showing a high correlation level (0.99 as the lowest value) and having the greatest differences in the later spike pairs. This could be caused by the inherent sampling error of the devices used. Nevertheless, the high correlation degree demonstrates that the TDE behavior does not change when it is deployed into an FPGA-based hardware in real time.

The resources needed by a single TDE unit were estimated for three different FPGA chips. In addition, the maximum number of TDE units that can be instantiated on each of them was reported. Table 4.9 summarizes all estimations.

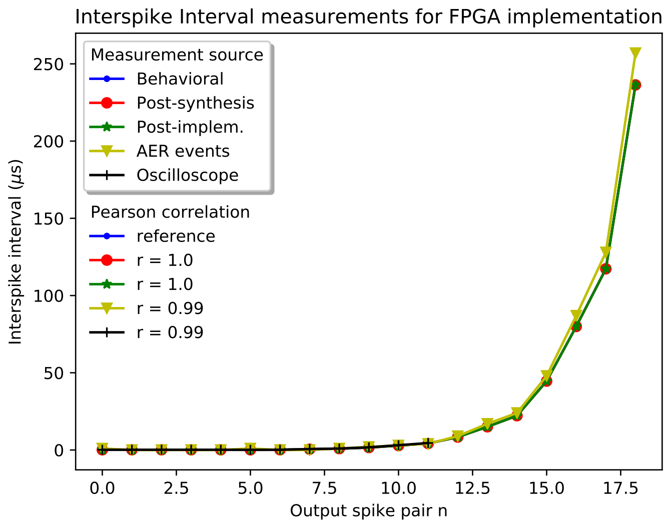


FIGURE 4.43: Comparing the TDE response from different measurement sources.

In addition, a high-level resource consumption comparison can be carried out between the analog implementation and the digital implementation. The former uses four capacitors for implementing the temporal decay of the signals, whereas the latter uses three timers instead. The difference lies in the absence of the refractory period in the proposed digital version.

Finally, a power consumption study was carried out for the XC6SLX150T chip, which was also used for all the measurements and real-time experiments in this work. Firstly, a set of Switching Activity Interchange Format (SAIF) files were used for a realistic and precise estimation, were a single TDE unit was

TABLE 4.9: Hardware resources utilization for different FPGA devices.

FPGA chip	Slice Registers Used / Available	Slice LUTs Used / Available	Max. num. TDE units
XC6SLX150T (Spartan-6)	122/184304 (0.07%)	207/92152 (0.23%)	445
XC7A75T (Artix-7)	140/94400 (0.15%)	179/47200 (0.38%)	263
XC7K480T (Kintex-7)	140/597200 (0.02%)	179/298600 (0.06%)	1668

stimulated with a simple pair of facilitatory and trigger events. The reported power consumption was less than 1 mW with a static power consumption of 98% (intrinsic to the FPGA). Then, the system power consumption was measured directly from the real hardware setup by measuring the power consumption in two different cases: 1) when the board was programmed and the reset signal was active (553.6 mW) and 2) when the board was programmed, the reset signal was not active, and an input events pair was sent (555.1 mW). Therefore, the measured power consumption for a TDE unit was approximately 1.5 mW, having a deviation of 0.5 mW with respect to the simulation estimation. These measurements were summarized in Table 4.10

TABLE 4.10: Power consumption summary of a TDE unit deployed into the XC6SLX150T FPGA chip.

Case to measure	Measurement
Board programmed and reset signal active (A)	553.6 mW
Board programmed, reset no active, input active (B)	555.1 mW
Measured power consumption for a single TDE unit (B-A)	1.5 mW
Estimated power consumption for a single TDE unit	1.0 mW

For comparison purposes, the power consumption of a single analog CMOS TDE implementation in the XFAB XP018 technology was estimated by means of a circuit simulation. The static power consumption amounts to 1.4 nW while the dynamic power consumption increases with the TDE's output spiking frequency, reaching approximately 500μW at 500 Hz.

4.3.4 Real-time neuromorphic application

The applicability of the proposed TDE model was evaluated by means of a proof-of-concept application. In the work presented in Milde et al. (Milde et al., 2018), the proof-of-concept was focused on a neuromorphic application using visual information generated by event-based cameras. In this thesis, a real-time sound source lateralization application for FPGA was implemented using the Neuromorphic Auditory Sensor (NAS).

Briefly defined, the sound source lateralization is considered as the capability to identify where the sound source is by using only binaural cues (Plenge, 1974). Different neuromorphic approaches to solve this task have been proposed in the last two decades (Finger et al., 2011; Park et al., 2013; Van Schaik et al., 2004; Schaik, 2010). Simplifying the concept, we will consider the sound source lateralization as the ability to determine whether the sound is on the left, on the middle, or on the right.

For this task, the same binaural cues that are commonly used for sound source localization (ITD and ILD) can be used. A binaural sensor is needed in order to be able to capture those cues. As introduced before, the Neuromorphic

Auditory Sensor (NAS), proposed in (Jimenez-Fernandez et al., 2017), is a neuromorphic sensor capable of decomposing the input sound from a pair of microphones into its frequency components, emulating the human cochlea. The general NAS architecture is depicted in Fig. 4.44 top. Since it is an event-based sensor, its output is encoded as events, thus the information is coded not only in the number of output spikes but also in the relative time between them. Only the ITD cue was used in the proof-of-concept application due to the timing nature given by the proposed TDE model.

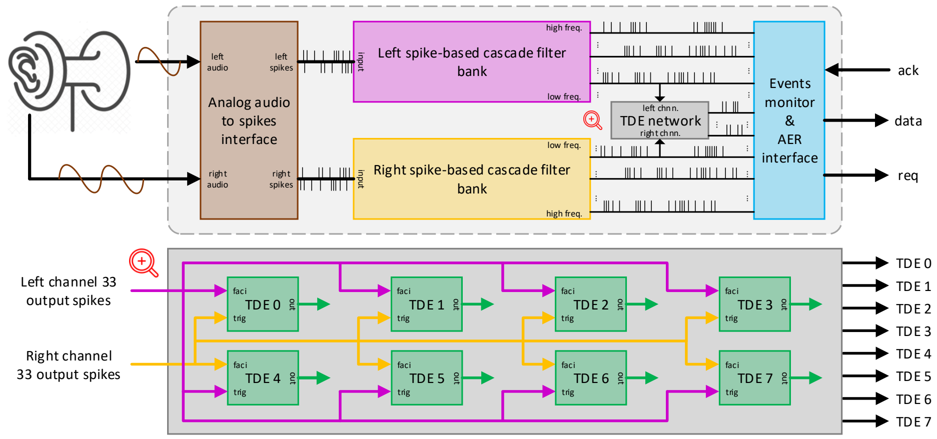

left audio
left spikes
Analog audio to spikes interface
right audio
right spikes
input
Left spike-based cascade filter bank
high freq.
low freq.
Right spike-based cascade filter bank
left chnn.
TDE network
right chnn.
Events monitor & AER interface
ack
data
req
Left channel 33 output spikes
Right channel 33 output spikes
faci
trig
out
TDE 0
TDE 1
TDE 2
TDE 3
TDE 4
TDE 5
TDE 6
TDE 7

FIGURE 4.44: Detailed block diagram of the FPGA top module for the proof-of-concept, containing both the NAS and the TDE populations.

The position of a sound source in space can be encoded by the temporal difference between the arrival of the sound waves at the ipsilateral side and the contralateral side. This time difference is known as the ITD. According to the specifications of the TDE model, an output response is exclusively produced if the incoming facilitatory event arrives before the incoming trigger event. Thus, two TDE populations were needed to perform the sound source lateralization task: one for detecting when the sound source is located at the left of the reference (the microphones pair) and one for detecting when the sound source is located at the right. Although the auditory information used was the same, thus containing the same temporal information, it projects onto the two TDE populations in an opposite way. The network architecture is shown in Fig. 4.44 bottom, and the parameters values used for the TDE units' configuration were the same as those presented in Table 4.8 for the microseconds time reference. Therefore, the individual tuning curves, as well as the population tuning curve, correspond to the plot shown in Fig.4.39.

The test scenario was designed as follows: firstly, a virtual room of 10 × 10 × 2 meters was created, using the RIR generator (Habets, 2006) software tool. A pair of directional microphones were placed in the center of the room, imitating the human's ears disposition in a regular head, and at a height of one meter over the floor. Then, three sound sources were placed at -90, 0 , and +90 degrees with respect to the microphones pair, corresponding to the left, front, and right positions, respectively, with a separation of two meters. The sound sources generated a pure tone beep of 500 Hz, with a duration of 0.5 seconds, every second.

Regarding the auditory sensor, a 64-frequency-channels, binaural NAS, with a frequency range from 22 Hz to 22 KHz was generated using the OpenNAS tool (Gutierrez-Galan et al., 2021b). The output events from frequency channel number 33 were used as input of the TDE population, since the center frequency of its associated event-based band pass filter was set to 502.38 Hz.

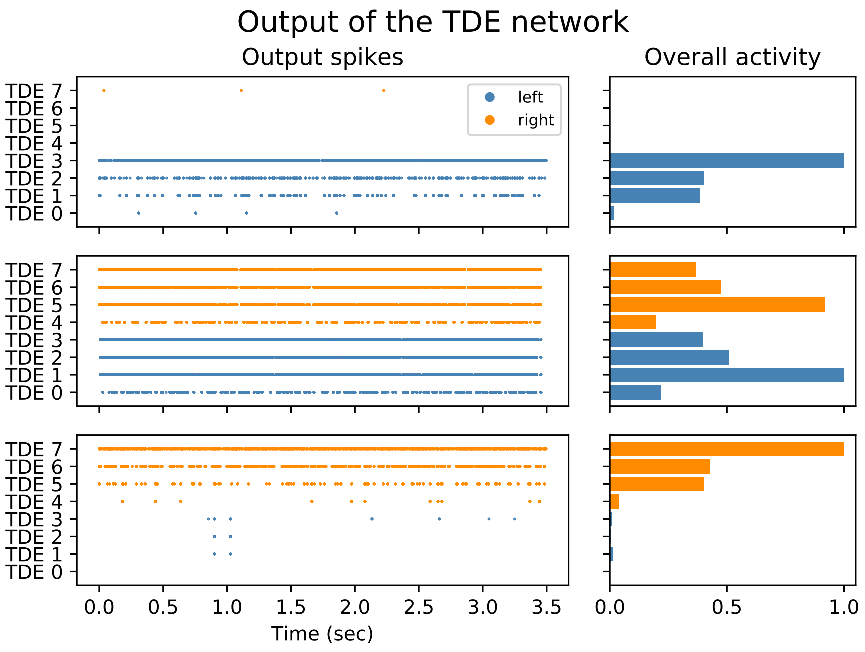


FIGURE 4.45: Raster plot of the output events from the TDE population and normalized overall activity. The plots were generated using pyNAVIS tool (Dominguez-Morales et al., 2021b). TDEs 0 to 3 (left population) correspond to indexes 0 to 3 in 4.8, and TDEs 4 to 7 (right population) also correspond to indexes 0 to 3 in the same table. Therefore, TDE 0 and TDE 4 use the same configuration and so on.

The sound was sent from the computer to the FPGA in real time, and the TDE population's output events were collected in a computer by using jAER. The results obtained from the real-time experiments are shown in Fig. 4.45. Top and bottom plots show the response of the TDE populations when the

TABLE 4.11: Comparison of a single neuron implementations using an FPGA. Zero slice registers used means that a memory was used to store the neuron state.

Paper reference	Neuron model	Modeling strategy	Temporal decay type	FPGA resources (LUTs/reg)
(Perez-Peña et al., 2019)	LIF	Biophysical emulation	Exponential	110 / 54
(Frenkel et al., 2017)	LIF	Phenomenological	Linear	90 / 0
(Frenkel et al., 2017)	Izhikevich	Phenomenological	Linear	247 / 0
(Soleimani et al., 2012)	Izhikevich	Biophysical emulation	Exponential	493 / 617
(Levi et al., 2018)	Hodgkin - Huxley	Biophysical emulation	Exponential	619 / 521
This work	TDE	Phenomenological	Linear	179 / 140

sound source was placed on the very left and very right positions (high ITD values), respectively. As expected, most of the activity was produced by the corresponding TDE population (left and right respectively). The maximum overall activity was found in those TDE units with higher detection time (TDE 3 for the left side and TDE 7 for the right side). On the contrary, TDE 0 and 4 barely presented any activity due to their low value for the detection time.

The center plot shows the response of both populations when the sound source was placed in front of the microphones pair. In this case, all the instantiated TDE units produced output events as a response to the low ITD value inherent to the input stimuli. The small asymmetry in the overall response can be explained by the fact that the I2S protocol samples the input sound sequentially. Therefore, left and right samples provided to the left and right cochleas as input are slightly different, thus producing different spike activity at the filter's output. This activity could be post-processed in order to extract more precise spatial information of the sound source.

4.3.5 Conclusion

This event-based design encodes temporal differences into a burst of events. A phenomenological design strategy was followed in order to implement the TDE behavior with reduced design complexity. The proposed implementation needs 179 LUTs and 140 registers. The comparison in Table 4.11 shows that the phenomenological design approaches use less FPGA resources since the design complexity is reduced.

The simulation data presented here faithfully reproduce the behavior reported in (Milde et al., 2018), where also a phenomenological analog implementation approach was followed. Once simulated, the model was deployed into an FPGA-based device in order to characterize its response in a real-time platform. Although the proposed model is also suitable to be implemented in a full custom ASIC, an FPGA-based platform was considered due to its reconfigurability, fast and affordable prototyping workflow. This allows creating custom TDE populations with different parameters and scaling up the population's size if needed, which is not possible in neither analog nor digital ASIC, unless specified at design time with high silicon overhead.

Another advantage of the FPGA TDE implementation is the wide temporal resolution range that it offers. It can be adjusted to different temporal domains by setting a few integer values depending on the specific applications, such as sound source localization (microseconds), vision (milliseconds), or odour localization (seconds). Since some neuromorphic systems such as the SpiNNaker (Furber et al., 2013a) board only provide a milliseconds resolution by default, the digital TDE supports the application to tasks which require a higher temporal resolution, as the sound source localization example provided in this work. Furthermore, the FPGA implementation does not suffer from the mismatch problem common in analog CMOS circuits.

However, it is also important to mention that, unlike the analog implementation, the model is not fully asynchronous. This feature forces the model to have clock signals, thus increasing the static power consumption due to the switching currents. Therefore, the reported power consumption of the proposed model is significantly higher (less than 1 mW) compared to the few μW of power consumption for low output spike frequencies (measured in CADENCE using XFAB XP018 technology) given by the subthreshold operation level in which the analog version works.

4.4 Comparison between both approaches

In both proposed models, a proof-of-concept of a sound source lateralization application were presented to evaluate the usability and performance of the proposed models with realistic input stimuli. The use of the TDE model for sound source lateralization represents an efficient alternative to the biologically-inspired Jeffress model (Jeffress, 1948). In fact, the TDE model may be more similar to the mammalian mechanism than the Jeffress model, which is specifically a model of the avian mechanism (Grothe et al., 2010). Since the TDE model encodes temporal differences in a frequency-coding manner, only two TDE's (one left, one right) are needed to encode the full range of sound source angles. However, the Jeffress model requires more neurons for extracting temporal differences, and the model's accuracy is proportional to the number of neurons.

This way, the hardware resources needed to implement this approach will be less compared to other state-of-the-art alternatives. Nevertheless, the combination of TDEs with different facilitatory time constants leads to an exponential population response, increasing the model's accuracy. In terms of post-processing steps, the Jeffress model already provides a clear position estimation from its output events, while the position estimation from the TDE's output needs to be extracted by applying further post-processing operations. Therefore, by comparing both models, it can be concluded that the Jeffress model is a more complex model but its output can be directly used as a position estimation, while the TDE model is less complex but its output has to be processed in order to extract the position estimation.

As was demonstrated in section 4.3.4, we can distinguish between at least three different sound source angles in the horizontal plane using the TDE's spiking frequency with four units for each side. Also, as showed in section 4.2.6, the Jeffress model is able to determine the position of a sound source within a virtual room with a high accuracy. Moreover, the output of both models could be send to other neuromorphic processors (e.g., Loihi (Davies et al., 2018) or SpiNNaker (Furber et al., 2013a)) to further improve the localization accuracy by using an SNNs.

In addition, thanks to the modular structure of both models, they can be adopted by the neuromorphic research community and seamlessly integrated with event-driven sensors to support the investigation of novel algorithms for bio-inspired sensing.

Chapter 5

Neuromorphic audio applications for robotics

"What is now proved was once only imagined."
– William Blake

5.1 Introduction

Initially, robots were developed with the aim of making our life easier, performing repetitive or dangerous tasks for humans. Although they were able to perform these tasks, the latest generation of robots is being designed to take a step further, by performing more complex tasks that have been carried out by smart animals or humans up to date. To this end, inspiration needs to be taken from biological examples. For example, insects are able to optimally solve complex environment navigation problems, and many researchers have begun to mimic how these insects behave (Haessig et al., 2020; Angelidis et al., 2021).

FIGURE 5.1: An octopus-inspired soft robotic arm. Credit: Harvard SEAS.

Another example is the foldable drone (Falanga et al., 2018), which is capable of changing its morphology to adapt it to the environment, thus being able to navigate through different spaces, in the same way that both pigeons and swifts adapt their wing surface. Recent interest in neuromorphic engineering (Bartolozzi et al., 2022; Christensen et al., 2022) has motivated researchers to take inspiration in animals not only to design the shape of the robot, but also to mimic the control mechanisms that manage the motor movement and sensors to collect external information and act in consequence.

Event-based cameras have been attached to mobile vehicles, such as drones (Maqueda et al., 2018) and wheeled robots (Milde et al., 2017) to collect information about the environment. Then, biologically inspired Spiking Neural Networks (SNNs) can be used to process this information due to the temporal features of the event-based data, and to control the robot's actuators using directly and uniquely events as input stimuli (Perez-Peña et al., 2013a).

The development of audio-based robotics applications has not followed the path of vision-based applications, thus finding fewer works where the audio processing and the robotics are mixed. Recent works in audio-guided robotics are related to the non-contact communication arising from the COVID-19 situation, as presented in (Grasse et al., 2021), where speech commands were used to control a robot to deliver objects without contact. Nevertheless, to the best of our knowledge, most of them do not use a neuromorphic sensor to collect the audio stimuli or a neuromorphic platform to process the collected data in real-time. The advantages given by neuromorphic audio sensors could be used to create a new generation of neuromorphic robots that includes a neuromorphic model of the hearing sense.

5.2 Motivation and cases of use

With the aim to demonstrate the use of the neuromorphic audio processing approach along with the robots and its advantages in real-time applications, a set of demonstrators were developed. For example, robot-human interaction could be considered one of the most exciting application in the future of neurorobotics, being an interesting topic when talking about bio-inspired solutions.

Even though we can consider the vision as the main source of information, it may happen that the region of interest is out of the field of view. Therefore, a second source of information would be desirable to identify and localize where the region of interest is and orientate the robot towards that position. This principle could be apply, for example, to vehicles for autonomous navigation or to humanoids robots to face the person that is talking. A real-time event-based sensory integration system was designed and implemented to demonstrate this concept.

It also could happen that the robot needs adapt the movement it is performing according to the auditory stimuli it is perceiving, like the animals are able to change their movement patterns depending on if they hear a prey or a predator. The brain area that recognizes the sound needs to be communicated with the area that generate the movement pattern to produce a fast reaction, that in some cases could be some milliseconds. A robotic platform was developed to implement, to the best of our knowledge, the first neuromorphic event-based audio-guided Central Pattern Generator (CPG) that is able to move up to twelve servo motors in real-time using a live connection between an FPGA and SpiNNaker.

Finally, all the concepts studied and developed during this thesis were intended to be deployed in one of the most complete robotics platforms in the world: the iCub humanoid robot. With this robot, the sensory integration, the attention, and the movement task can be studied together in order to study and understand how humans carry out some specific tasks. In addition, it could help to test the new generation of auditory devices before using them in humans, and also to better understand some neurodevelopmental disorders, such as autism, and its relation to the hearing sense.

5.2.1 NeuroPod: from audio to locomotion through spiking CPG

A Central Pattern Generator (CPG) is a neural structure located at a spinal cord level. It can generate rhythm patterns which might be used for movements, such as the generation of various gaits, or swimming (Grillner et al., 2008). There is proven evidence of such structures in small animals (Duysens and Van de Crommert, 1998) and possibly in humans (Guertin, 2009; Minassian et al., 2017). The activity of these structures is released and mediated by the brain stem and other sub-cortical regions of the brain. Regarding the feedback, the CPG receives sensory information to adapt its output to the environment.

Locomotion is probably one of the most complex tasks to be developed by roboticists due to stability issues when several legs are involved (Schilling et al., 2013). Therefore, from a neurorobotics point of view, the idea of these CPGs is borrowed from biology to implement locomotion in small robots with several legs. The reason for this is that these structures can generate a very stable pattern even without sensory information or brain activity. In fact, some cats that suffered severe spinal cord injuries, recovered their gaits after treadmill training sessions (Vogelstein et al., 2008). Real-time biomimetic CPGs research is also promising for connecting to ex vivo spinal cord with the aim of controlling neuroprosthesis in hybrid experiments (Ambroise et al., 2013).

There are many works where a CPG has been used within robotics; some of them mimic the idea of a CPG, although without implementing a spiking neural network. Instead, they modelled the CPG using differential equations of coupled oscillators. Examples are: (Sartoretti et al., 2018), where the authors used a hexapod robot and they included feedback, (Barron-Zambrano et al., 2010), where the Van der Pol oscillator model was used and implemented on a FPGA, and (Crespi et al., 2008a), where the authors used a swimming and crawling fish robot and implemented the CPG on a microcontroller by solving the equations of coupled oscillators.

More closely related works, where neuromorphic hardware was used or a SNN was proposed, are: (Still et al., 2001), where the authors developed an analog neuromorphic dedicated chip which allocates coupled oscillators and a learning procedure to have the desired output, although they did not use well-known neuron models, and (Donati et al., 2014), in which the authors designed and implemented several CPGs segments to drive a lamprey-like robot. The CPGs were implemented using neuromorphic hardware in (Qiao et al., 2015), although online changes of the pattern generated by the CPG are not provided. Likewise, the work presented in (Cuevas-Arteaga et al., 2017) proposed the implementation of the CPG using a SNN implemented on SpiNNaker (Furber et al., 2014b), although it does not offer real time nor online change of the pattern produced by the CPG. Table 5.1 summarizes the main information about the CPG and real-time capabilities of the systems presented in some of the state-of-the-art studies that have been discussed in this section.

In this section, a CPG closely related to its biological counterpart, including plausible biological features, was implemented in SpiNNaker (Gutierrez-Galan et al., 2020). In addition, the CPG model was extended for behaving according to external auditory stimuli through the Neuromorphic Auditory Sensor (NAS). Both models were ready to be used within robotics, and a hexapod robot was used to validate the designs and to show the main novelties of this research, which are the real-time operation of the CPG and the online reconfiguration of the gait produced by the CPG.

5.2.1.1 The hexapod robot

The NeuroPod robot is divided into three main parts, and each of these has a specific role or functionality. These parts are the CPG, designed using a SNN and implemented on a neuromorphic hardware platform. This CPG generates the gait patterns. The movement controller takes the movement information from the CPG and controls a set of servomotors through an FPGA-based board. Finally, the skeleton defines the shape of the robot and also performs the movements. Further details are provided in the following sections. Figure 5.2 shows a global overview of the NeuroPod as a block diagram; it also shows the main parts and how they interface with each other.

An hexapod robot is a six-legged robot inspired by arthropod insects, such as ants or flies, among others. According to biology, the body of these insects can be divided into three different regions: the head, the thorax and the abdomen.

TABLE 5.1: Comparative study between state-of-the-art approaches regarding CPG-based systems

Ref.	Spike-based	Materials and methods	Real-time operation	Number of gaits	Online transition between gaits
(Ambroise et al., 2013)	Yes	- Implements a leech heartbeat neural network on FPGA	Yes	1	-
(Sartoretti et al., 2018)	No	- Coupled oscillators as CPG. - Modified to allow adaptation to the terrain	Yes	1	No, but parameters can be modified
(Barron-Zambrano et al., 2010)	No	- Non-linear coupled oscillators implemented in a soft-core processor for FPGA	Yes, but only done in simulation	3	Yes
(Crespi et al., 2008a)	No	- Chain of non-coupled oscillators	No	1	No, but parameters can be modified
(Still et al., 2001)	No	- Coupled oscillators as CPG using a novel neuromorphic chip - Support Vector learning algorithm	Yes	1	-
(Donati et al., 2014)	Yes	- Segments of the lamprey CPG - A mixed signal analog/digital VLSI device interfaced to an FPGA - Spikes are routed to/from a standard PC through the FPGA	Yes	1	-
(Cuevas-Arteaga et al., 2017)	Yes	- Oscilator-based CPG implemented on SpiNNaker - SpiNNaker generates local file with results - Arduino reads local file and controls servomotors	No	3	No
(Rostro-Gonzalez et al., 2015)	Yes	- Oscilator-based CPG implemented on an FPGA	Yes	3	Yes

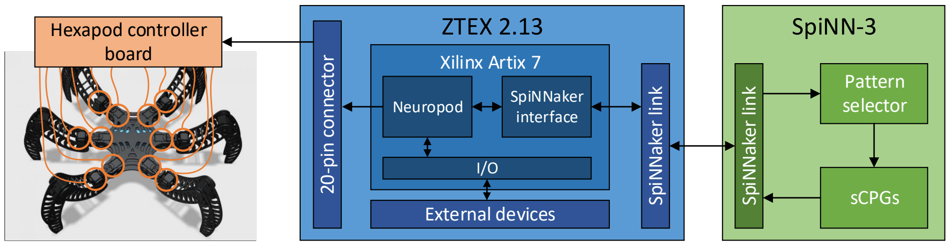


FIGURE 5.2: Block diagram of the entire system. It is composed of the SpiNNaker board, an FPGA-based board, and a 3D-printed hexapod robot frame.

Moreover, each part could be sub-divided into segments according to their features.

The head is composed of eyes (located in the ocular segment) and a pair of antennae. Both the eyes and the antennae are used to collect sensory information about the environment, allowing the movement of the insect in complex scenarios by performing an obstacle avoidance task (Douglass and Strausfeld, 1995; Milde et al., 2015). The thorax is composed of three segments: the prothorax, the mesothorax and the metathorax. Each segment has a pair of legs, and there are six in total. Up to five parts can be identified in each leg, although only three of these parts are relevant to motion: coxa, femur, and tibia. Finally, the abdomen contains the vital organs of the insect, such as the respiratory or reproductive systems.

Recent focus on the development of smart robots by mimicking biological processes has motivated many research groups to develop accurate models of

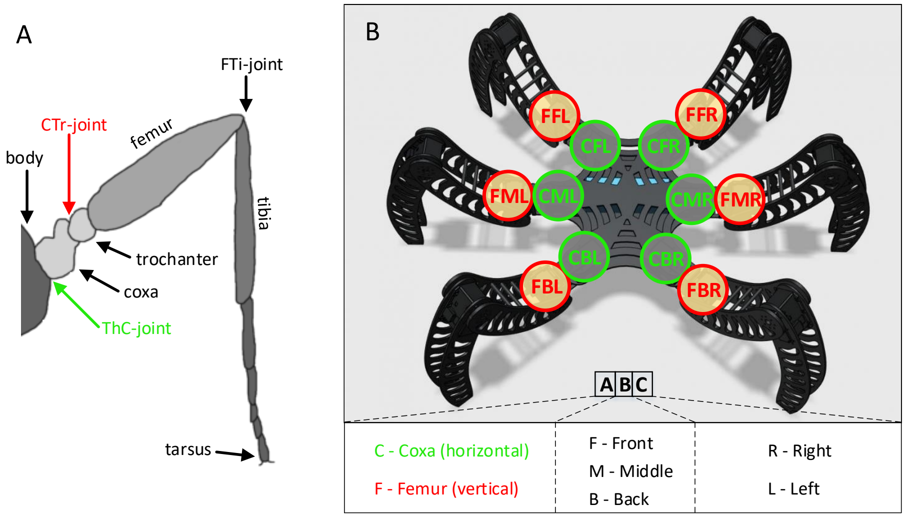


FIGURE 5.3: A) Biological representation of an arthropod's leg anatomy. B) Hexapod robot leg actuator IDs.

hexapod insects. HECTOR (Schneider et al., 2012) is an example of that, where both the body features and movements were inspired by the morphological details of the stick insect *Carausius morosus*.

In this work, a 3D-printed hexapod robot was used, based on the model featured in (Cuevas-Arteaga et al., 2017). The original design[1] was adapted by designing a new body frame to allocate the electronic devices on it. The frame dimensions are $20mm \times 89mm \times 90mm$ (height, width, depth), without the legs.

According to (Büschges et al., 2008), insect legs are defined as multi-segmented limbs. Each leg consists of more than 5 segments, as it is represented in Fig.5.3A. However, only three of them are used when performing a movement: the coxa, the femur, and the tibia. This is due to the fact that three main leg joints can be found in an insect leg: the thoraco-coxal (ThC-) joint, the coxa-trochanteral (CTr-) joint and the femur-tibia (FTi-) joint. The ThC-joint is responsible for carrying out back and forth movements (horizontal axis), the CTr-joint enables elevation and depression, and the FTi-joint allows extension and flexion (both in the vertical axis).

The leg of each hexapod has three degrees of freedom (DOF), one per joint. However, to develop NeuroPod we only considered two of them because the movement of the robot can be performed mainly using the coxa and the femur (Rostro-Gonzalez et al., 2015). Thus, only twelve DOF were taken into account, instead of eighteen, to implement the gait patterns.

[1]https://www.thingiverse.com/thing:1021540 (checked on July'2022)

In order to provide motion, one servomotor was placed on each joint, making a total of twelve servomotors (Ref. SG90, https://servodatabase.com/servo/towerpro/sg90). The maximum rotation angle is 180 degrees, although this range could be reduced due to the mechanical constraints of the body design and the position of the servo on it.

Therefore, a previous calibration is required. After the calibration process, and knowing that the theoretical operation speed of the selected servos is 0.12 s/60 degrees, we were able to estimate the pattern period, which can be defined as the minimum time the robot needs to reach the backward position, starting from the forward position, and then reach the forward position again. Measurements of these pattern periods are presented in section 5.2.1.5.

5.2.1.2 Hardware setup: bi-direction communication between an FPGA and SpiNNaker in real-time

As it was mentioned in Section 1.2.2, the SpiNNaker project is based on a massive parallel multicore computing system that is able to run very large SNNs in real time (Furber et al., 2013b). The architecture of the SpiNNaker chip, which has an asynchronous packet switching network, makes it very efficient for neuromorphic applications (Plana et al., 2007).

In this work, the SpiNN-3 machine (4 SpiNNaker chips, 72 200MHz ARM9 cores) was used to implement the SNN model, which is described in section 5.2.1.3. The device is shown in Fig. 5.4. This board has an interface, 100 Mbps Ethernet link, which is used to control the SpiNNaker machine from the computer. It also has two spinn-link connectors that enable a connection to external devices such as FPGAs and neuromorphic sensors: retinas or cochleas. This board was connected to an FPGA for real-time bidirectional input/output communication through the spinn-link interface. This interface is based on the 2-of-7 protocol. Since most of the neuromorphic sensors use the AER protocol to communicate with other devices, an AER-SpiNN VHDL module was developed by the SpiNNaker group (Plana et al., 2014), which converts from AER protocol to 2-of-7 protocol, and vice versa. This module was, in addition, adapted to the project requirements by our research group[2].

An FPGA-based board was used to implement a digital system design, which controls the neuromorphic robot platform. This approach was considered in other similar works to implement a hardware version of CPGs, such as (Barron-Zambrano et al., 2010) and (Rostro-Gonzalez et al., 2015), and also in the field of neurorobotics and neuromorphic engineering (Yousefzadeh et al., 2017). This reconfigurable hardware offers flexibility against analog designs and adaptability in real time, in case of system failures.

[2]https://github.com/dgutierrezATC/NAS_SpiNNaker_interface

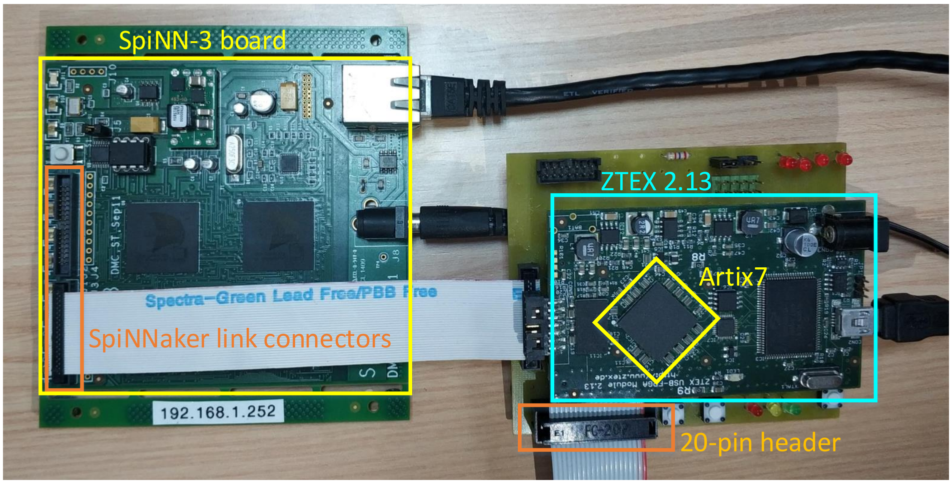


FIGURE 5.4: SpiNN-3 machine and ZTEX 2.13 board.

From the Xilinx Artix-7 family, the XC7A75T chip was used, mounted on the ZTEX 2.13 USB-FPGA board with a 48 MHz clock source. This FPGA chip offers around 75500 logic cells, 100 GPIOs, USB 2.0 interface and DDR3 SDRAM memory. This board serves as a daughter board mounted on a custom PCB provided with several components: LEDs, user buttons, an AER 20-pin interface and a spinn-link interface (see Appendix C). Those interfaces will be used by the SpiNNaker machine to manage the hexapod robot through the FPGA board. An extended explanation is provided in section 5.2.1.2.

5.2.1.3 Spiking Central Pattern Generator

A CPG is a neural network in which interconnected excitatory and inhibitory neurons produce an oscillatory, rhythmic output as a motion pattern, such as walking, flying, running or swimming, with the absence of rhythmic inputs. In this work, we focus on three specific gaits: walk, trot and run, which are selected based on previous working bio-inspired implementations for hexapods (Rostro-Gonzalez et al., 2015; Cuevas-Arteaga et al., 2017).

Fig. 5.5 (bottom) shows the basic structure for each of the CPGs implemented in this work. It consists of eight Leaky Integrate-And-Fire (LIF) neurons: two neurons for the SCPG and six output neurons to command the servomotors. The green and red neurons (Fig. 5.5) make the other neurons, ranging from 0 to 5, fire within different timings generating the selected gait. Each of these six neurons are then connected to two output neurons which will command two different servomotors. This is achievable due to the symmetry of the robot: pairs of servomotors always perform the chosen gait, independently of the selected

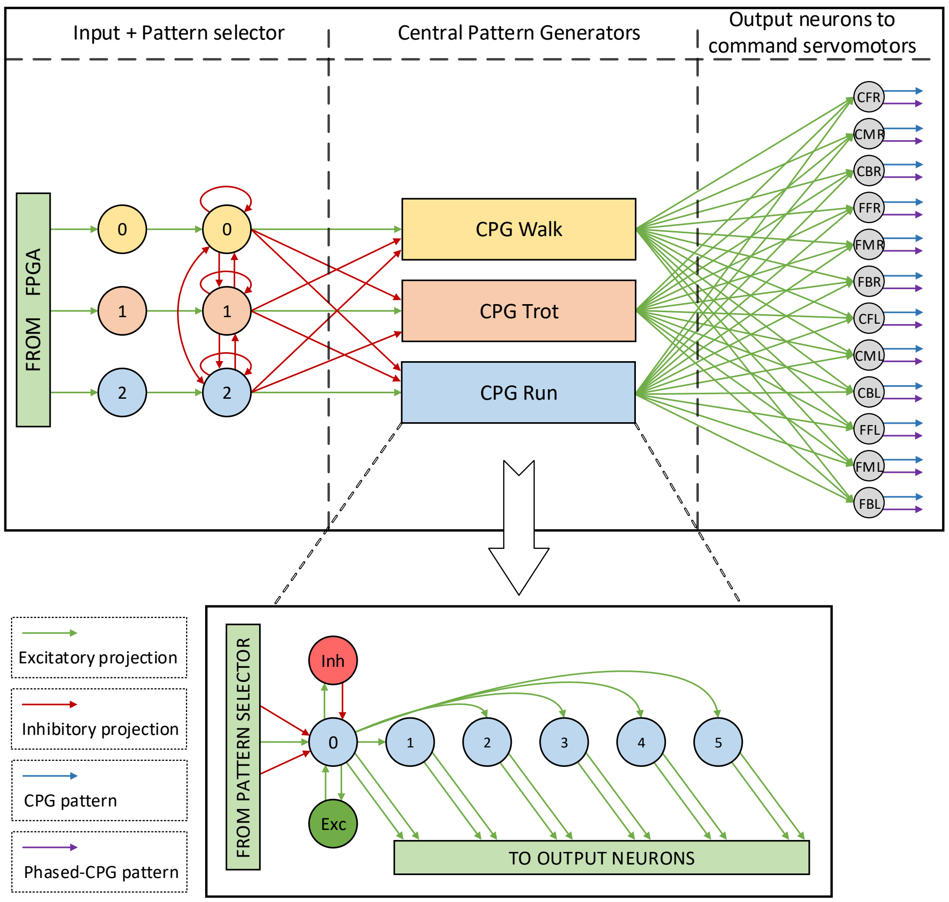


FIGURE 5.5: Diagram of the spiking neural network model used (top) with an in-depth view of the CPG architecture (bottom).

SCPG. Table 5.2 presents the parameters of the LIF neuron model that has been used in this work.

Three SCPGs following this basic architecture (one per gait) are enclosed within a global SNN model shown in Fig. 5.5 (top). This global network acts as a mechanism to select which of the SCPGs has to be enabled in order to start generating the gait, inhibiting the other two at the same time. It receives a single spike from the FPGA with a specific neuron address (0, 1 or 2) (using the AER protocol) indicating the gait pattern that needs to be generated. For this, we used a custom PCB (see Fig. 5.4) to connect the FPGA and the SpiNNaker board through the spinn-link connector for bidirectional I/O communication. When this spike reaches the pattern selector population in the SpiNNaker board

TABLE 5.2: LIF neuron parameters.

Parameter	Description	Value	Unit
cm	Capacitance of the LIF neuron	0.25	nF
i_offset	Base input current to add each timestep	0.0	nA
tau_m	Time-constant of the RC circuit	20.0	ms
tau_refrac	Refractory period	2.0	ms
tau_syn_E	Excitatory input current decay time-constant	5.0	-
tau_syn_I	Inhibitory input current decay time-constant	5.0	-
v_reset	Voltage to set the neuron at immediately after a spike	-68.0	mV
v_rest	Ambient rest voltage of the neuron	-65.0	mV
v_thresh	Threshold voltage at which the neuron will spike	-50.0	mV

(the three neurons that are closer to the SCPGs), this group of neurons transmit this spike to the selected SCPG, while inhibiting any other that could have been running at that moment, and also make that SCPG (associated with the gait that was selected) start generating the spikes. This mechanism allows real-time gait changes by activating the appropriate SCPGs without introducing a long delay (hundreds of ms), which is essential for real-time robotics applications.

A single spike is needed by the selected SCPG to start generating the spiking pattern. The spikes fired by the six SCPG neurons are sent to the last layer of the model, which consists of twelve neurons corresponding to each of the hexapod servomotors. These spikes are transmitted back to the FPGA using the spinn-link, where a circuit commands each of the servomotors using the live output spikes.

As is shown in Fig. 5.5, each neuron of the output layer has two outputs from SpiNNaker to the FPGA. The first one is the regular spiking pattern generated by the SCPG to command the servomotors (extension action). The second one is exactly the same pattern, but phased with a delay of 1 tick, needed by the FPGA to command the servomotors back to the standard position (performing the flexion action).

5.2.1.4 NeuroPod HDL top module architecture

As previously mentioned in section 5.2.1.2, a digital system is needed to implement the neuromorphic robotic controller. This task is often carried out by using an FPGA-based board, running a custom digital system. Designs are generally implemented using VHDL, which allows defining any digital circuit model by describing either its behavior or its components' interconnections. Each component of the design is also known as a module, and many functional modules can be encapsulated by a top module, which defines both the input and output signals of the digital circuit.

Fig. 5.6 shows an overview of the proposed implementation of the NeuroPod control system top module. It performs three main functions: to select the gait

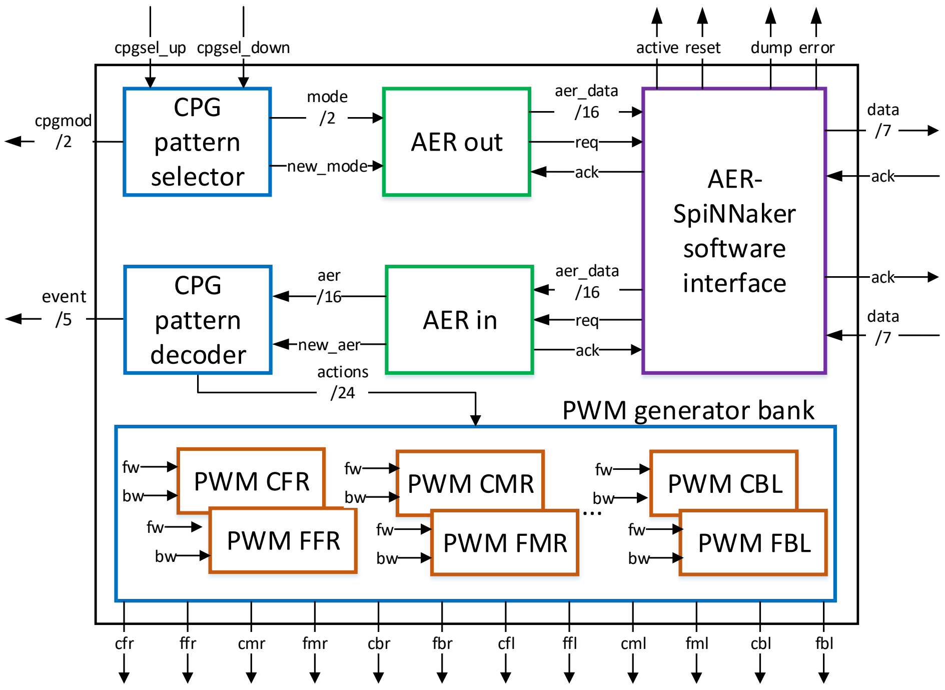


FIGURE 5.6: NeuroPod FPGA top module overview.

which the SCPG implemented on SpiNNaker will generate, to send the gait information to the SpiNNaker machine, receive the live output pattern from it and, finally, to generate the Pulse-Width Modulation (PWM) signals to control the servomotors. Further details are given next, starting with the pattern selector and then following accordingly to the work flow.

First, the CPG pattern selector module was implemented as a 2-bit up/down unsigned counter. Both up and down signals are declared as input and they are directly mapped to two buttons located on the base board in which the ZTEX board is connected. The current counter value indicates the gait: 0 for walking, 1 for trotting, and 2 for running. This information is shown to the user by means of a pair of LEDs in binary format.

Every time the Central Pattern Generator (CPG) pattern selector changes its value, an interruption is generated through the signal *new_mode* to notify that there is a new data available to the next module. This component is the AER out module, which converts a 2-bit value to a 16-bit AER event, and also handles the AER handshake protocol (*REQUEST* and *ACK* signals). This conversion to AER is carried out since the system needs to send information to the SpiNNaker to generate the pattern.

TABLE 5.3: AER decodification scheme.

	CFR	**FFR**	**CMR**	**FMR**	**CBR**	**FBR**	**CFL**	**FFL**	**CML**	**FML**	**CBL**	**FBL**
FW	0	1	2	3	4	5	6	7	8	9	10	11
BW	12	13	14	15	16	17	18	19	20	21	22	23

As it 5.2.1.2, a VHDL module was developed by the SpiNNaker team (Plana et al., 2014) for interfacing AER-based neuromorphic devices with the SpiNNaker machine. It takes AER events as input following the AER protocol, and generates packets under the 2-of-7 protocol for the communication from the AER device to the SpiNNaker board. For the reverse communication, it takes 2-of-7 packets and generates AER events. In addition, this module provides four status signals to check in real-time if the communication is working properly.

AER events received by the SpiNNaker, generated by the AER-SpiNN module, are captured by the AER in module, which implements the handshake and sets the value of the event as a 16-bit output signal. In the same way as the AER out module, an interruption is enabled every time the AER-in module receives a new input event.

Those input events correspond to the spikes fired by the output layer neurons of the SNN implemented on SpiNNaker. Since each output neuron manages the position of one servomotor, these events have to be mapped to the correct one. Therefore, depending on the address of the event, an action is performed over a servomotor.

There are two actions available: either move the servomotor towards the forward position or move the servomotor towards the backward position. The decoding scheme is summarized in Table 5.3, where each column represents the joints of the NeuroPod following the same nomenclature as in Fig. 5.3: the **FW** row means **forward action**, the **BW** row means **backward action**, and each value is the AER event which triggers the action.

The commands received are converted to motion through a PWM generator block, which receives the decoded AER events by means of a 24-bit signal (one enabling signal per action). This PWM generator block was implemented instantiating as many PWM generators as the number of joints that the NeuroPod has.

The PWM generator implemented includes some features that make the NeuroPod motion control easier: up to three pulse width values can be set in the same VHDL module instead of only one. These values were used to define the positions that the servomotor had to reach. Those positions are: forward, backward (corresponding to the actions), and home. Two control signals were added to the PWM generator to select the configuration of the module: *fw*, which enables the generation of the PWM signal associated to the FW position, and *bw*, which enables the generation of the PWM signal associated with the BW position.

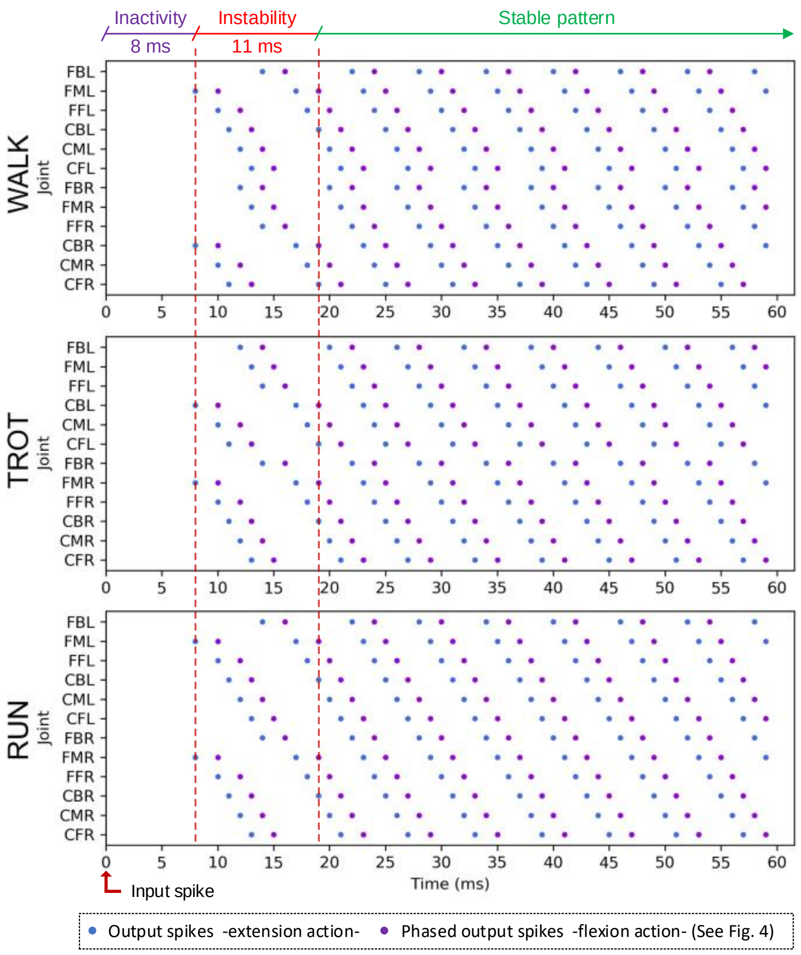


FIGURE 5.7: Output spikes for each gait pattern simulated on SpiNNaker.

This module has two input signals, which are connected following the scheme shown in Table 5.3. When a control signal is set to high, either through a single pulse or a constant signal, the pulse width value associated with that control signal is loaded in the configuration register. Then, the PWM output signal changes automatically, moving the servomotor to the commanded position. That output signal is held until the module receives a different action command. Finally, the home position is only activated when the global reset signal is released.

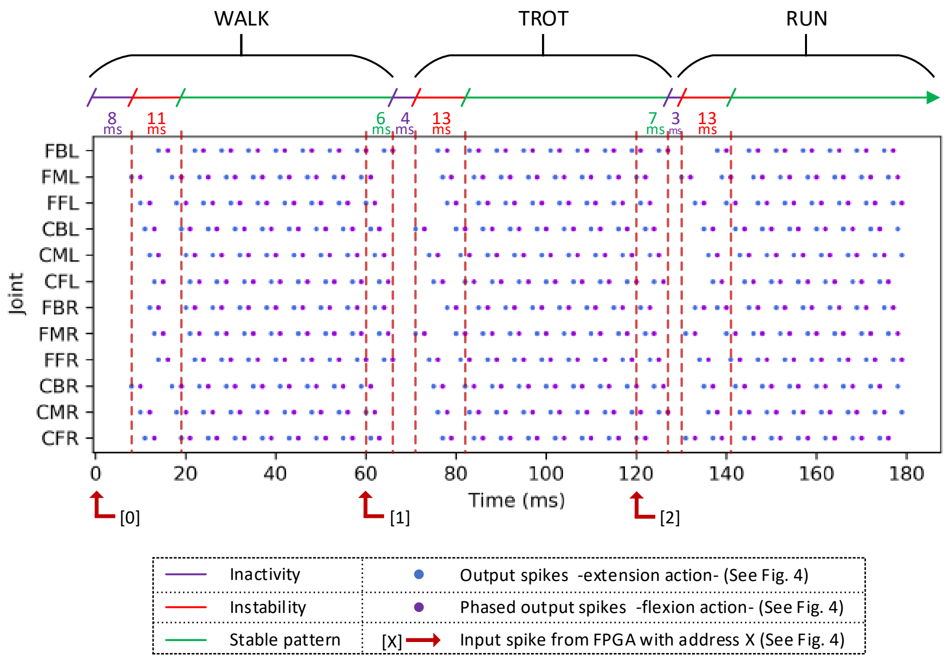


FIGURE 5.8: Output spikes from the SpiNNaker simulations that show the gait pattern change behavior.

5.2.1.5 Simulation results

The results obtained for the simulation of each of the gait patterns on SpiNNaker are shown in Fig. 5.7. This figure shows the output spikes that the SCPG generates.

Then, in Fig. 5.8, the same plot is shown for a different scenario in which we simulated and tested the behavior of the SNN when forcing the system to change from a specific SCPG to a different one. The figure shows how the system is able to change from walk to trot and then to run, generating a stable pattern for each gait after a specific period of time, which, in this case, is 23 milliseconds. This delay is the time that the network takes to inhibit the neurons related to the previous gait that was being executed plus the time that the neurons related to the current pattern take to start generating the correct firing output in a stable way. Different delays related to these simulations were measured and are presented in the image.

It is important to mention that these delays were measured in simulation. For a real-time scenario, the delays do not match these values since, due to the fact that the servomotors were not able to work at such speed, we had to set the time_scale_factor parameter on the SpiNNaker board to 100. This made the whole simulation run 100 times slower in real time, which is very convenient for this approach.

Regarding the FPGA, a study of the VHDL module was carried out. A post-implementation resources consumption report was generated, and also post-implementation simulations were performed to measure the delays of every single VHDL module. The obtained results from those analyses are shown in Table 5.4. To obtain these times, a clock source of 48MHz (23.83 ns of clock period) was used.

TABLE 5.4: FPGA resources consumption and delays

	Resources consumption		**Delays**
	LUTs	Registers	Time (Clock cycles)
CPG pattern selector	2 (<0.01%)	4 (<0.01%)	20.83 ns (1)
AER out	7 (0.01%)	5 (<0.01%)	104.15 ns (5)
AER-SpiNN interface	213 (0.45%)	272 (0.29%)	1374.78 ns (66)
AER in	7 (0.01%)	10 (0.01%)	41.66 ns (2)
CPG pattern decoder	12 (0.03%)	24 (0.03%)	20.83 ns (1)
PWM generator block	720 (1.53%)	576 (0.61%)	1895.53 ns (91)
NeuroPod Top	**986 (2.09%)**	**893 (0.95%)**	**3457.78 ns (166)**

The amount of resources used by the top module is around 2.1% of the available LUTs and around 1% of the available number of registers. Most of the resources were consumed by the PWM generator due to the fact that it needs a clock divisor, a square signal generator, and a duty-cycle counter for each PWM signal to be generated. Therefore, the synthesis tool infers three counters along with its combinational logic control circuitry. This low resources consumption will allow us to implement more complex spike-based motor control modules (Jimenez-Fernandez et al., 2012; Perez-Peña et al., 2013b) as well as improving the NeuroPod top module by including both visual and auditory input information about the environment.

In addition, the delay added by the full design, in the worst case, is almost 3.5 µs, which is irrelevant compared to the delays presented in Fig. 5.8.

After the simulation results were obtained, a real-time analysis of the full system was carried out. Both the latency from a high level command to the generation of the CPG and the actual motion of the leg were measured using an oscilloscope. These delays can be discarded since they are three orders of magnitude lower than the time taken by the SpiNNaker to generate the SCPG which is 800 ms (the simulated time updated with the time_scale_factor).

According to that latency and the time that a gait cycle takes to be performed, we can conclude that the theoretical maximum value of the movement speed is 4.16 cm/s. This value is obtained from the distance travelled by the hexapod divided by the time taken to do it. If we look at the WALK gait, it can be seen that all the legs (6 legs in total) take 12 milliseconds to complete a cycle (with a cycle being the period of time that the hexapod takes until its legs start repeating the same pattern again).

Considering that every single motor moves 30 degrees each time, the following equation can be used: $distance\_traveled = sin(angle\_moved) * femur\_length$. Replacing the values: $distance\_traveled = sin(30) * (5) = 2.5cm$ This value is multiplied by two since the complete cycle is done by the two sides of the hexapod. Then, the 12ms should be multiplied by the $time\_scale\_factor$ (100) resulting in 1.2s. If now, we divide the distance travelled by that time, the maximum speed is the result: 0.05cm/1.2s = 4.16 cm/s. That will offer the maximum theoretical speed. The general equation is:

$$Speed = \frac{femur\_length}{time\_scale\_factor * time\_gait} \tag{5.1}$$

After that, we compared the simulation time with the real-time delays[3]. To this end, four cases of study were defined: resting to movement, stabilization time, movement's period, and change time between two different gaits. There were time differences since the parameter time_scale_factor was set to 100 in the SpiNNaker script in order to slowdown the SpiNNaker output.

The delays introduced in the open-loop control are very low, in the order of ms (see Table 5.4). Previous works have a 50ms delay of propagation (Donati et al., 2014), or a convergence time of 5 seconds in (Crespi, Ijspeert, et al., 2006). Our results show a time of 23 ms (worst case) to converge and around 20ms to propagate the gait. Another difference with (Donati et al., 2014) is that we propose to use SpiNNaker (Furber et al., 2014a) instead of the neuromorphic chip ROLLS (Qiao et al., 2015). Also, the CPG proposed in this work uses 30 neurons (considering the pattern selector and the CPGs) which is 10 neurons less than (Donati et al., 2014).

[3]Demonstration video is also available: https://youtu.be/YZYAPDJHvLI

5.2.1.6 Towards an audio-guided behavior

Changes in animals' locomotion are motivated, in general, by either looking for food or avoiding predators. In both processes, external stimuli are received through sensors like the eyes, the skin, or the ears. The collected information about the environment is then processed by the brain, thus producing a reaction to that input stimulus. This reaction is translated to movement by the spinal cord, as it was explained in Section 5.2.1.3. Applying this idea to robotics, and using the advantages of neuromorphic systems, a new generation of robots with learning capabilities and low-power consumption is starting to gain popularity.

For instance, a fish robot with light, touch, and water sensor (Crespi et al., 2008b) were used for feeding a SCPG in order to study the adaptation processes, in real-time, in the robot's behavior when unpredictable external stimuli were present. In addition, sensors' output data was used as feedback to the network for improving the locomotion, achieving a closed-loop system. Similar work was presented in (Youssef et al., 2020), where a fish robot was equipped with a pair of frame-based cameras and a pair of event-based cameras. The main task is to identify an underwater light stimulus and, depending on its color, activate the SCPG locomotion to either follow (simulating a prey) or avoid (simulating a predator) the light source in real time.

The event-based cameras were also mounted in an hexapod robot (Ting et al., 2020; Lele et al., 2021), using the visual stimulus for learning the locomotion by imitation and also for object tracking. All of these works used a SNN for implementing the SCPG in real-time, including adaptation modules for providing the system with certain autonomy. Although visual stimuli could be considered as the main source of information, auditory information becomes crucial when vision does not provide relevant information.

In this direction, some works can be found in the literature. Gomez et al. proposed in (Gomez-Rodriguez et al., 2007) a system where a CPG, implemented in VHDL for FPGA, was managed by the output of an analog cochleae (Yang et al., 2016) for moving a humanoid robot. The cochlea's output was processed in order to extract the beats for the event's burst, modifying the CPG's output for each detected beat. Furthermore, a review of the state-of-the-art in the use of the peripheral auditory system of lizards implemented in biorobotics models for sound source localization was carried out in (Shaikh et al., 2016).

Most of the implementations use SCPG models, as well as sensors for interacting with the system and to cause a change in the output locomotion in real time. However, the use of a fully neuromorphic system (together neuromorphic sensors and neuromorphic platforms for deploying the SNNs) is not that usual. Therefore, the proposed NeuroPod system was updated in order to include a neuromorphic auditory sensor for changing the output locomotion of the robot (Gutierrez-Galan et al., 2019a). The aim was to implement a real-time sound recognition system that, depending on the input sound, the hexapod robot react

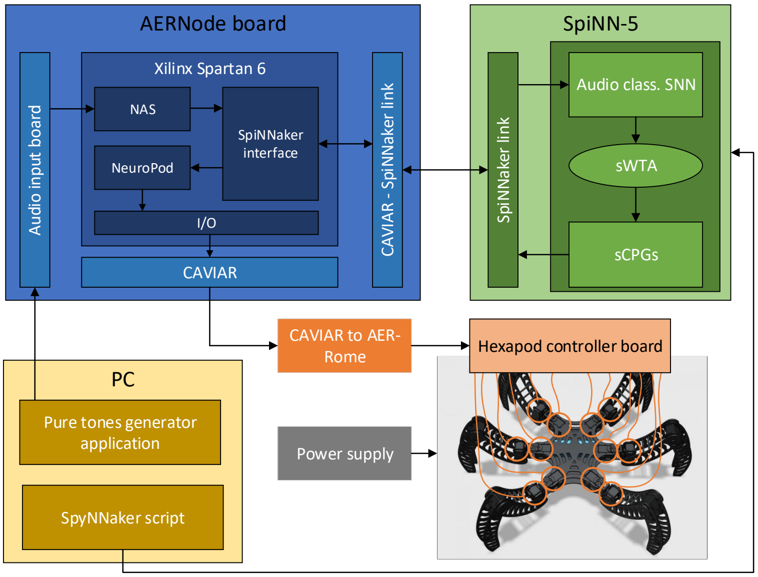


FIGURE 5.9: Block diagram of the updated NeuroPod system. It is composed of the SpiNNaker board, implementing different networks, an FPGA-based board, in which the NAS was deployed, and a 3D-printed hexapod robot frame.

in a specific way, similar to the work presented in (Youssef et al., 2020). An overview of the new NeuroPod system is shown in Fig. 5.9.

The sound recognition system, already presented in (Dominguez-Morales et al., 2016) and in (Dominguez-Morales et al., 2017a), was composed by a NAS and a multilayer SNN deployed on SpiNNaker, and it was used to classify eight different pure tones in real time. The training method of the SNN was designed by the authors based on the NAS output, achieving 100% of accuracy without any noise, and decreasing that value until 74% when the SNR values was set to 3 dB. The network was composed of three layers: the input layer, that receives the events from the NAS, and it has as many neurons as NAS' frequency channels; the hidden layer, that extracts features from the input sound, and it has as many neurons as the number of pure tones to be classified; and the output layer, that provides the output classification by mean of the output firing rate, meaning that the neuron that fires the most is the winner. This layer also has as many neurons as tones to be classified (eight in this case), and its output is used as gait selector.

In the FPGA side, the update was carried out by replacing the CPG pattern selector module (shown in Fig. 5.6) to a mono NAS with 64 frequency channels, as shown in Fig. 5.10, in the FPGA top module. The button input signal to the top module was replaced by the new input interface, which were the I2S protocol

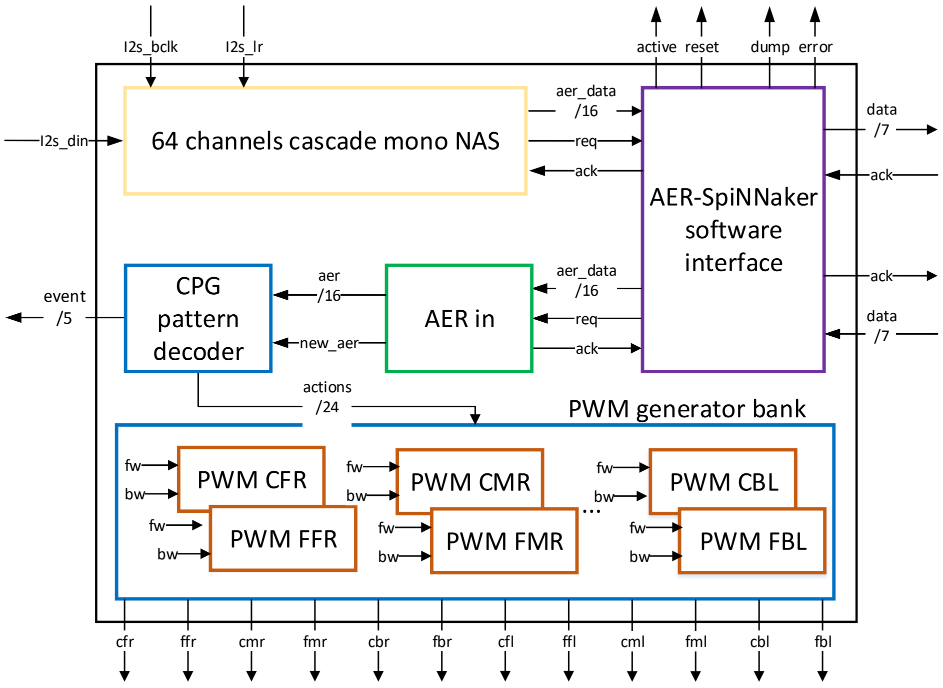


FIGURE 5.10: Audio-guided NeuroPod FPGA top module overview.

signals for sending the audio samples to the NAS. No other VHDL module was neither replaced nor modified thanks to the modularity of the design.

Regarding to the SCPG original network, shown in Fig. 5.5, several modifications were needed. First, the SNN for the pure tone classification was added, without any change, to the original SNN, thus having now two well-differentiated parts. Then, the output layer of the pure tone classification network was connected with the input layer of the SCPG network, where the pattern selector is located. Since the original SCPG network was able to generate up to three different gaits, only three output neurons from the pure tone classification network (neurons' IDs 0, 2, and 4) were connected one-to-one with the input layer of the SCPG with an excitatory connection.

In addition, a new neuron was added to the input pattern selector layer of the SCPG network to stop the gait generation for stopping the robot's movement. This neuron has an one-to-one excitatory connection with the neuron 6 of the pure tone classification network, and a one-to-one excitatory connection with a global inhibitory neuron. This global inhibitory neuron is able to inhibit the whole SCPG network, having an inhibitory connection with a connection architecture one-to-all.

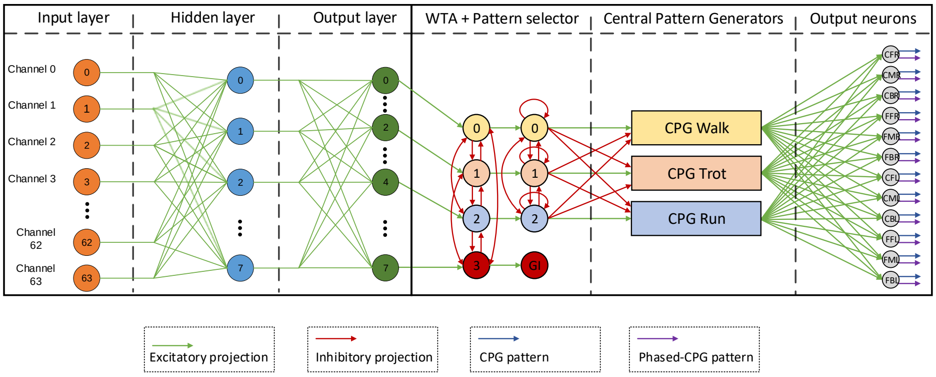


FIGURE 5.11: Diagram of the complete spiking neural network model used for the audio-guided NeuroPod. This network is composed of several sub-networks, where each of them has a specific task.

The architecture of the input layer was slightly modified since the gait selection now will come from another SNN, the sound detection network, instead of from the button on the FPGA. This layer was converted to a WTA network, thus modifying the parameters of both the neurons and the synaptic connections, in order to adapt the neurons to the new input firing rate. In the original work, this process was carried out by software in the host computer since there was not any further processing. A delay population network was added to each SCPG output neuron for slowing down the SCPG's output spikes in order to be able to move the robot's motors. In the original work, this adjustment was performed by setting the running options. However, since the pure tone classification network was trained for running with the normal configuration, the delay population was needed to adapt the SCPG output timing. The final network's architecture is shown in Fig. 5.11.

Finally, the hardware setup was also adjusted for allowing to run the new system's configuration. The SpiNN-3 machine was replaced by the SpiNN-5 machine for allowing the deployment of bigger networks due to its 48 SpiNNaker chips. Moreover, the FPGA board was also changed since different connectors were needed. The AER-Node board (Iakymchuk et al., 2014) was used instead of the ZTEX 2.13. Fig. 5.12 shows the entire hardware setup of the updated version of NeuroPod. A GUI was implemented for facilitating the script running and the pure tone generation through four buttons corresponding to the four pure tones that were used. This software application also receives and shows the output classification in order to check if the pure tone recognition network works properly.

A set of tests were carried out for checking the transitions between the patterns' generations. The main goal of these tests was to check the behavior

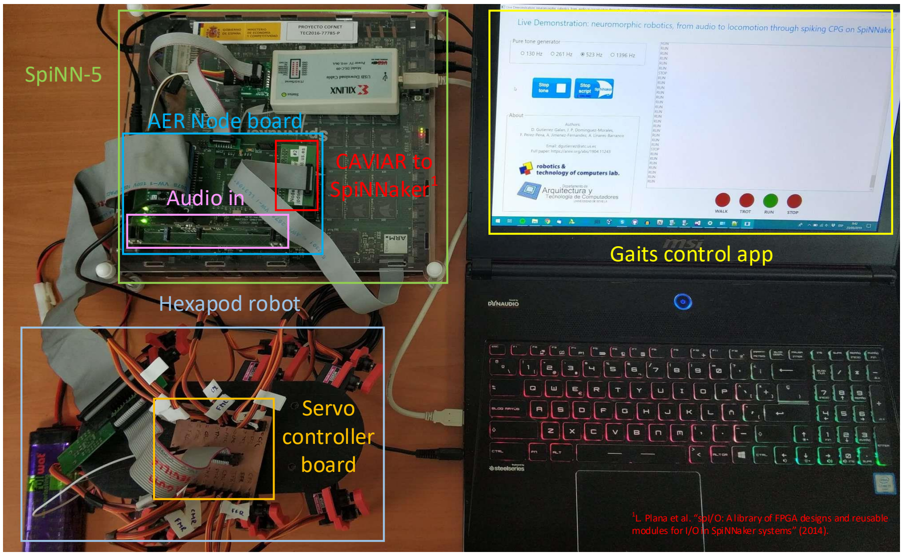

SpiNN-5
AER Node board
CAVIAR to SpiNNaker[1]
Audio in
Hexapod robot
Servo controller board
Gaits control app
[1]L. Plana et al. "spI/O: A library of FPGA designs and reusable modules for I/O in SpiNNaker systems" (2014).

FIGURE 5.12: Audio-guided NeuroPod hardware setup.

of all the networks during the gait's transition and to evaluate the result of the transitions. It may happen that the recognized sound does not match with the input sound, thus producing the wrong gait, or even that the generated movement does not follow any expected pattern due to a failure in the firing synchronization. Some factors which could cause these errors are: to use a different volume on the computer for playing pure tones compared to the one used for the training process, or to detect several input stimulus changes in a short time period, among others.

Fig. 5.13 shows the running output for one test case. Three different plots were represented: the output events of the input layer (top), the WTA (middle), and the output layer (bottom). The input layer corresponds to the neuron population that receives the events from the external NAS, while the output layer corresponds to the SCPG output that is used to move the motors. The test comprised four different parts. Firstly, the system was started, and no input sound was played. Secondly, a pure tone of 261 Hz was played for two seconds. Thirdly, the pure tone was changed to 130 Hz and played again for two seconds. Finally, the frequency of the pure tone was changed again to 1396 Hz, and it was played until the end of the experiment, being the test's duration ten seconds.

According to the input stimuli, the expected output behavior is the following: if no input stimulus is presented, there should be no output response; if the 261 Hz pure tone is played, the SCPG network should produce the TROT pattern; if the 130 Hz pure tone is played, the output pattern should be WALK; and when the 1396 Hz pure tone is played, the SCPG network should stop

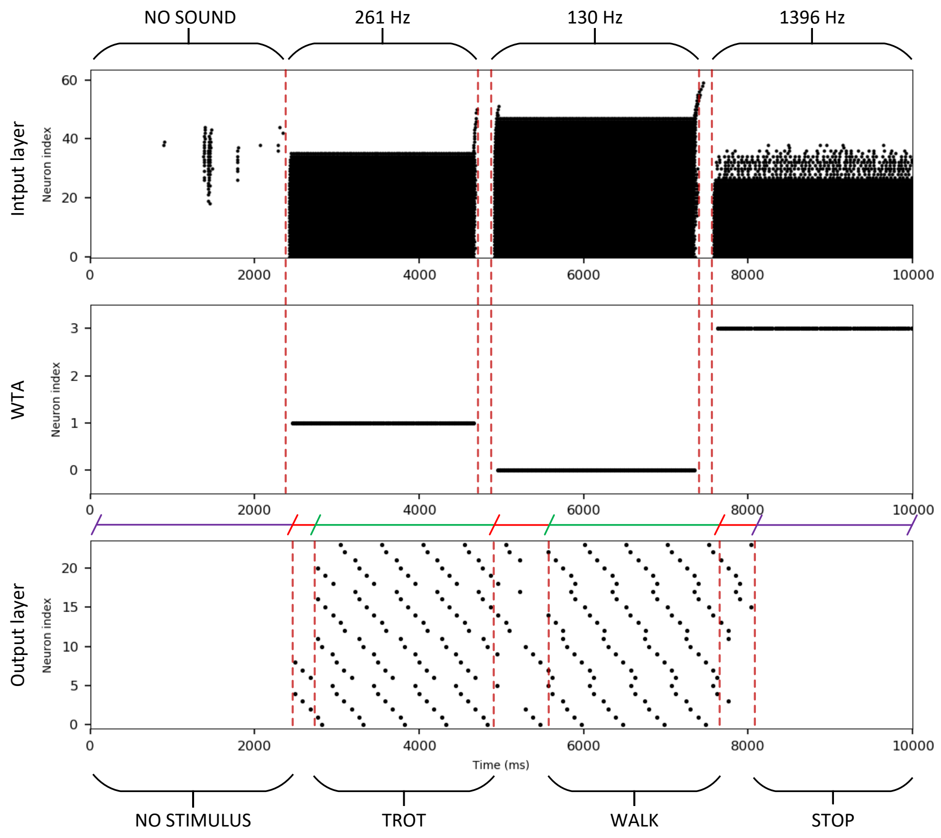


FIGURE 5.13: Output spikes from the SpiNNaker simulations that show the gait pattern change behavior according to the input auditory stimuli.

producing events. It can be observed in Fig. 5.13 that the obtained output matches with the expected behavior. No output events were produced in the absence of input sound. In addition, the sound recognition network achieved a 100% of accuracy in the pure tone classification, as it is shown in the output of the WTA network.

After the first auditory stimulus is received and recognized by the network, there is a short period of instability until the TROT gait is correctly generated. This effect was already shown in 5.8. The time delay between the firsts auditory events and the firsts movement's events is just a few tens of milliseconds, and a few hundreds of milliseconds for the instability part. Then, the TROT gait is properly generated until the frequency of the pure tone changes. The empty spaces in the input layer plot are produced by the software that generates the pure tones since it first stops the tone generation and then produces the new one. After changing the tone's frequency, and the WTA identifies a new winner, the pattern selector network stops generating the current pattern and starts generating the new one, thus having again an instability period. This effect also occurs when the last pure tone is generated. However, in this case the recognized pure tone corresponds to the STOP command, therefore not producing any output events.

This proof-of-concept shows that the idea of having a neuromorphic audio-guided robot can have multiple applications, such as rescue missions after natural disasters or checking tasks in dangerous places. The research will be focused on to learn how the auditory information is processed and how it could interact with locomotive decisions using neuromorphic hardware applied to neurorobotics.

5.2.2 Audio-visual sensory integration

Living beings combine various sensory cues with previously acquired knowledge to safely navigate towards a target destination. In close analogy to biological systems, a neuromorphic system which decides how to reach a sound source without collisions, based on auditory and visual input, was designed and implemented. The development of this sensory integration system, which identifies the shortest possible path, is a key achievement towards autonomous robotics.

In the presence of acoustic stimulation alone, the heading direction points to the direction of the sound source. When a visual input is introduced into the network, the heading direction always points at the direction of null optical flow closest to the sound source. Hence, the sensory integration network is able to find the shortest path to the sound source while avoiding obstacles. This section shows that a simple, task-dependent mapping of sensory information can lead to highly complex and robust decisions. The proposed neuromorphic system comprises two event-based sensors (the eDVS for vision and the NAS for audition) and the SpiNNaker processor.

This work (Schoepe et al., 2019) was carried out in collaboration with Thorben Schoepe and Elisabetta Chicca, from the CITEC at Bielefeld University and the Bio-Inspired Circuits and Systems (BICS) Lab at the University of Groningen. The optical flow encoder network was developed as part of Thorben Schoepe's thesis, while the sensory integration network is the result of the research collaboration between our research groups, and it belongs to both theses.

5.2.2.1 Problem to solve

Collision free navigation in a cluttered environment requires fast and robust decision making. Animals take decisions in a timescale of tens of milliseconds to execute collision avoidance (Barron et al., 2015). Furthermore, they take more complicated decisions based on multimodal sensory information combined with previously acquired knowledge. For example, the female budgerigar (a small Australian parrot) incorporates auditory and visual input to track down a male. The bird uses auditory cues, the Interaural Level Difference (ILD) and the Interaural Time Difference (ITD), to estimate the male's position (Amagai et al., 1996). While approaching the male, the female effectively avoids collisions thanks to visual information. First investigations indicate that the bird merges Optical Flow (OF) information with other visual cues to avoid obstacles (Altshuler and Srinivasan, 2018).

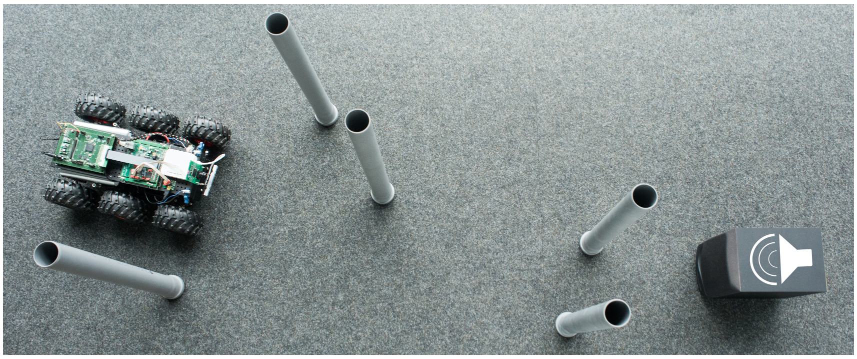


FIGURE 5.14: The autonomous robot navigates through a cluttered environment. It tries to reach a sound source while avoiding obstacles.

A few task specific SNNs which combine different sensory cues and previously acquired knowledge have already been proposed. (Chan et al., 2012) and (Finger et al., 2010) increase the localization preciseness of their sensory integration network by merging different sensory cues which point at the same target. (Horiuchi, 2009) combines previously acquired knowledge with one sensory cue to reach a target direction without collision. We present a new type of SNN which mimics the behaviour of the budgerigar and other animals.

5.2.2.2 Bio-inspired solution

The proposed bio-inspired network is able to identify and follow the direction of a sound source while avoiding obstacles. The model consists of an Optical Flow Encoder (OFE) network and a Sound Source Direction (SSD) network which receive sensory input from the embedded Dynamic Vision Sensor (eDVS) (Lichtsteiner et al., 2008) and the Neuromorphic Auditory Sensor (NAS) (Jimenez-Fernandez et al., 2017) respectively. The two networks feed into the Sensory Integration (SI) network which chooses the agent's heading direction. We evaluate the network's performance in open loop by applying different combinations of auditory and visual stimuli to the two sensors.

5.2.2.2.1 Optical Flow Encoder Network (OFE)

The speed of an object moving in the visual field of a translationally moving agent is inversely proportional to its relative distance. Bees, flies, and some bird species use this visual cue called Optical Flow (OF) to safely navigate through densely cluttered environments (Altshuler and Srinivasan, 2018), while humans use the OF to balance. Since the discovery of OF, various OF encoding algorithms and models have been designed (Brosch et al., 2015), based on the Hassenstein-Reichardt-Detector (Hassentstein and Reichardt, 1956).

One very recent OF detector model is the sEMD proposed by Milde et al. (Milde et al., 2018). It encodes the time difference between two spikes from adjacent pixels in both the number of output spikes and the Interspike Interval (ISI). In this work we feed filtered data from the eDVS into a sEMD population to encode the spatial distribution of OF (see Fig. 5.15). A spatio-temporal filter population between the eDVS and the sEMDs reduces the noise and the spatial resolution of the visual information (See (Milde et al., 2018) for further information). The OFE network's output provides topographically arranged relative distance information in form of OF to the SI network. The whole OFE network was implemented on SpiNNaker.

5.2.2.2.2 Sound Source Direction Network (SSD)

Birds use the ILD and the ITD to perform the sound source localization task (Amagai et al., 1996). While ILD achieves better accuracy with high-frequency sounds, ITD performs better when low-frequency sounds are present(Liu et al., 2013). As it was detailed in Section 1.3.1, the ITD can be estimated by calculating the correlation between the input stimuli from both ears to determine the position of a sound source.

According to (Jeffress, 1948), the correlation can be calculated by using an array of spike-based coincidence detector neurons. The excitatory output spikes from the cochlear nucleus Cochlear Nucleus (CN) are fed into those detectors through delay lines. Depending on the sound source position, the sound waves arrive earlier to one ear than the other. Those input stimuli coincide in a specific coincidence detector, which identifies the estimated position. Note that the time difference is directly related to the distance between the ears.

Since the ears' distance in birds amounts to just a few centimetres, time differences are in the tens of microseconds range. These fine temporal delays are not calculable on SpiNNaker due to limitations in temporal resolution. Because of that, a spike-based Jeffress model was designed as a real-time VHDL module to be added along with the NAS, detailed in Section 4.2. However, the coincidence detector neurons project onto a WTA network (Oster et al., 2009) implemented on SpiNNaker to decrease the noise in the SSD network's output. The WTA network feeds spikes into the Lateral Sound Transmitter (LST) population explained in the next section (see Fig. 5.15).

5.2.2.2.3 Sensory Integration Network (SI)

The OFE network's retinotopic output map and the SSD network's tonotopic output map are arranged topographically. Both networks project (directly or indirectly) onto the SI network's Decision Making Winner-Take-All (DMWTA) map (see Fig. 5.15). This type of mapping seems to be quite efficient since it has been found in different vertebrates which have been optimized over millions of years (Kaas, 1997). The different sensory maps have to be correctly aligned to each other to combine multimodal sensory cues in one network.

In human beings the spatially more reliable visual input teaches the adaptation of the auditory input map (Shi and Müller, 2013). Such an alignment adaptation has been simulated in neuromorphic systems (Chan et al., 2012; Finger et al., 2010). Given the current open loop configuration, there is no need for an adaptive alignment of the visual and auditory maps. Therefore, we simply map the corresponding positions in all three networks.

Besides the alignment, the importance of different sensory information with respect to decision making has to be taken into account. Visual information always dominates the proposed network since collision avoidance is an essential task to guarantee the agent's damage-free navigation. To achieve that, the OFE network's output (visual information) strongly inhibits the SI network's DMWTA population (see Fig. 5.15). This guarantees that the agent never drives into the direction of a nearby object because the DMWTA population's output defines the heading direction.

Whenever the visual field is object free, the heading direction equals the sound source direction created by the SSD network. That means that the SSD network's output could directly be mapped onto the DMWTA population. Still, when an object appears directly in the sound source direction the corresponding DMWTA neuron is strongly inhibited so that it can not win. In this condition the LST population comes into play (see Fig. 5.15). The LST neuron positioned at the SSD excites the two adjacent neurons. This lateral excitation further spreads through the LST population until a position with zero OF input is reached. At that position the DMWTA population releases a spike.

Since the DMWTA population consists of a hard WTA network (Oster et al., 2009) the winning neuron inhibits all other decision making neurons. At each instant the network can only decide for one specific heading direction. The selected heading direction always appears at the position of null OF closest to the sound source direction. This is caused by the fact that the lateral excitation wave in the LST population reaches the closest position with null OF with the smallest delay and with the highest excitation. The lateral excitation decreases with increasing lateral sound source distance given that the lateral connections are weak.

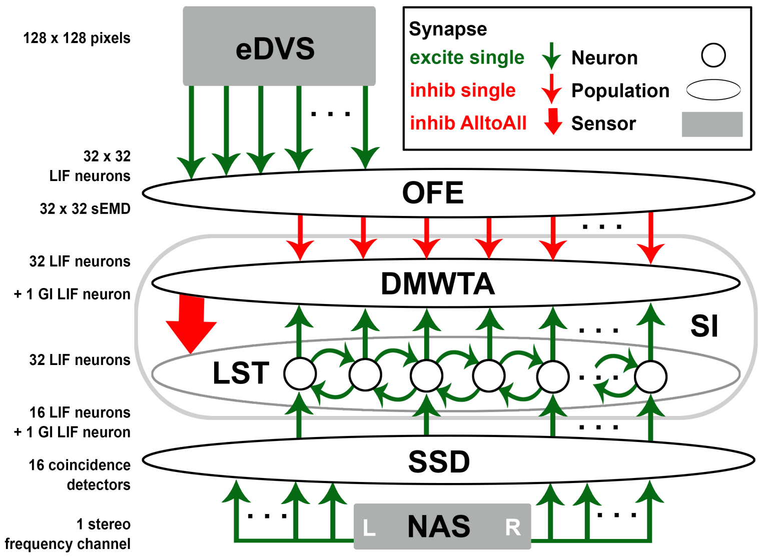


FIGURE 5.15: Complete network. The OFE population consists of the SPTC LIF population and the sEMD population. The SSD population includes coincidence detector neurons and an additional hard WTA network. Each WTA consists of an excitatory LIF neuron population and one GI neuron. All excitatory LIF neurons are connected to the GI neuron. The GI neuron projects back onto the excitatory LIF neurons.

This WTA structure matches with findings in mammals supporting the hypothesis that competing alternatives switch off each other through inhibition (Barron et al., 2015). When a spike is released by the DMWTA population, it also sets back the LST population by inhibition.

5.2.2.3 Hardware setup

The AER DVS128 retina chip (Lichtsteiner et al., 2008) comprises pixels which mimic the bipolar cells present in the mammalian retina. It consists of an array of 128×128 independent pixels that respond to relative light intensity changes in real time and are intrinsically invariant to scene illumination. A pixel produces an event in response to a change in luminance over time. As soon as the event is produced, the address of the pixel (x and y coordinates, and polarity) is written on an arbitrated handshaked asynchronous bus known as the Address Event Representation (AER) bus. The eDVS (Müller and Conradt, 2011) consists of a DVS128 chip connected to an ARM microcontroller. This device is intended for embedded robotics.

The NAS (Jimenez-Fernandez et al., 2017), that was already detailed in Section 1.3.2.1.4, is a spike-based audio sensor inspired by Lyon's model of the

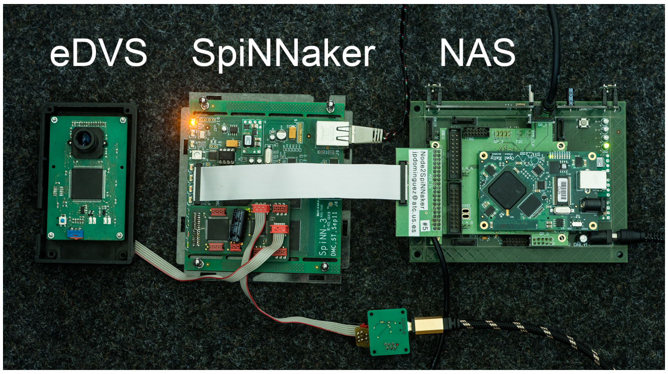


FIGURE 5.16: eDVS and NAS feeding spikes into a SpiNN-3 board through the SpiNNlink connectors. The eDVS can directly process visual input while the NAS receives stereo audio input through an audio jack.

biological cochlea (Lyon and Mead, 1988), implemented on FPGA. This sensor decomposes incoming audio signals in their frequency components as the inner hair cells do in the inner ear. It was implemented using a SLPF bank with a cascade topology (Jimenez-Fernandez et al., 2010). Each SLPF represents a frequency range, and its output consists of a stream of AER events. In this work, we used a 64-channel binaural NAS generated with OpenNAS (Gutierrez-Galan et al., 2021b) [4]. In addition, a 4-node SpiNNaker machine was used since it has two SpiNNaker links for connecting two external devices. Those interfaces were used to connect the eDVS and the NAS as input to the SNN (see Fig. 5.16).

5.2.2.4 Simulation test and results

Two experiments were conducted to characterize the network's performance. In the first experiment, only auditory information was fed into the network to verify that the SI network's heading direction follows the sound source direction. In the second experiment, OF was added to investigate the network's behaviour when it tries to follow a sound source in a cluttered environment.

In the first experiment, a synthetic audio file was generated by using a Python script along with the RIR generator library. In this script, a 500 Hz sound source was swept from left to right at 2 meters receptor distance. This recording was fed into the NAS in order to check the SI network's sound source following behaviour.

For all tests, events from one of the 64 NAS channels with a center frequency close to the sound source frequency of 500 Hz were used. As shown in Fig. 5.17 (a), the heading direction (identified as neuron id) follows the sound sweep from

[4] https://github.com/RTC-research-group/OpenNAS

left (high id) to right (low id). The correlation between expected and achieved heading direction amounts to 89% (Pearson correlation coefficient) (Benesty et al., 2009).

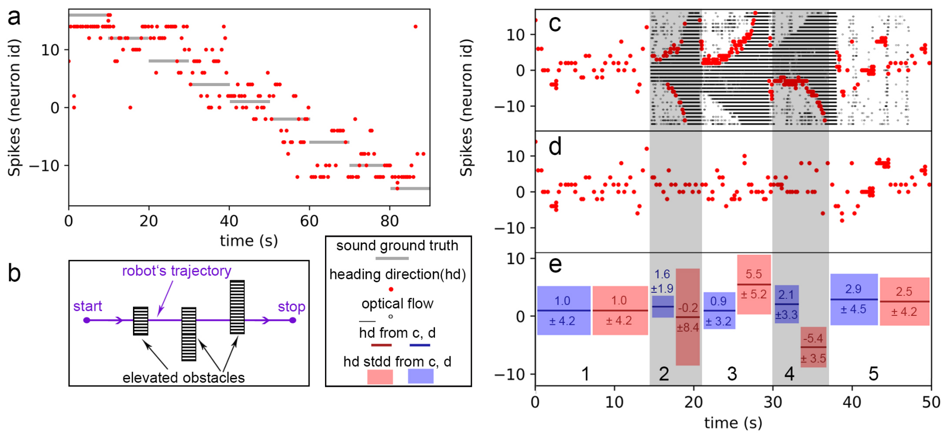


FIGURE 5.17: (a) SI network's heading direction response to a 180 degrees sound source direction sweep. (b) Setup to record OF data used in Figure 5.17c. An eDVS is mounted on top of a robotic platform which drives purely translational with a speed of ~0.8 m/s through the scene. The first obstacle is located in the middle of the visual field, the second one on the right side and the last one on the left side. All obstacles are positioned at least 40 cm above the ground so that the robot can drive underneath them. Heading directions for a centered sound source with OF (c) and without OF (d). (e) Heading direction mean and heading direction standard deviation (stdd) with OF (red) and without OF (blue) for five different time periods with different visual scenarios

In the second experiment, a synthetic audio file with a centered sound source generated similarly as in subsection 5.2.2.2.2 was fed into the NAS. Additionally, eDVS recordings were projected into the OFE network. These recordings were done with a robot executing pure translational motion through the environment as shown in Figure 5.17 (b).

As long as only sound information is fed into the whole network, the mean of the heading direction lies as expected close to neuron id zero, which corresponds to the sound source direction (Fig. 5.17 (d) and (e)). The same accounts for region one in Figure 5.17 (c) and (e) because the robot is not moving. In Figure 5.17 (c), in region two, three, and four, a high amount of OF changes the heading direction. In region two, the heading direction's standard deviation is very high. As explained in subsection 5.2.2.2, the heading direction always points to the direction of null OF closest to the sound source direction.

Since the obstacle is located in the middle, there is no clear closest direction with zero OF and the heading direction fluctuates a lot between both sides. This could be seen as a problem, but in case of a closed-loop experiment, the first laterally located spike will cause a turn of the robot so that the object is not centrally located anymore. What happens in case of a laterally positioned object can be seen in region three. The heading direction points significantly to the left.

This can be explained by the fact that the obstacle is located at the right side. This makes the path at the left side around the obstacle the shorter one. In region four, the same effect can be shown but with the obstacle on the left side. After avoiding the obstacles, the heading direction goes back to the middle. This is almost identical to the behaviour without OF (Region 5). As expected, the SI network always points at the direction of null OF closest to the sound source.

The proposed sensory integration SNN shows the expected behaviour: it adjusts its heading direction to the sound source direction with a correlation of 89%. When OF is introduced into the network the heading direction always points at the direction of null OF closest to the sound source. Hence, the sensory integration network is able to find the shortest path to the sound source while avoiding obstacles under well-defined test conditions.

5.2.2.5 First steps to a closed-loop system

After testing the system using recordings from both sensors and demonstrating the ability of the network to select the shorter path around obstacles toward the sound source, real-time input stimuli were feed into the SNN. In addition, the system was improved by closing the loop. Two different test scenarios were taken into account: 1) real-time input with static sensors' positions and 2) real-time input with dynamic sensors' positions.

In the first test scenario, a NAS, a 3DIO ears-like microphone[5], a eDVS, and a SpiNN-3 machine were used, as shown in Fig. 5.18. The 3DIO microphone was used as audio input for the NAS, which was directly connected to SpiNNaker. A headphone was placed over the microphone to avoid external noise and to control the input sound stimulus. In addition, the eDVS sensor, also connected to SpiNNaker, was placed on a table and the background objects were removed.

For the auditory stimulus, three positions of the 500 Hz pure tone were used: -90º, 0º, and 90º (left, middle, and right). This way, the sound source always belongs fixed to a specific position in the space. For the visual stimulus, the optical flow generation was carried out by shaking an object in front of the eDVS since it was fixed on the table. This way, an obstacle could be simulated by shaking the hand for a time in front of the camera or in one of both sides, as shown in Fig. 5.18. No ego-motion events are induced due to the eDVS is fix on the table, thus facilitating the obstacle detection from the OF.

The SNN model was slightly adjusted for receiving real-time events that may contain some noise compared to the events from the recordings. Fig. 5.19 shows

[5] https://3diosound.com/products/free-space-binaural-microphone

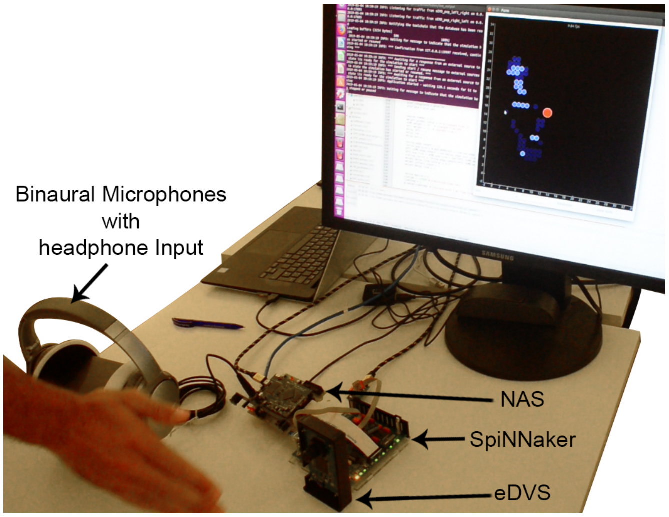


FIGURE 5.18: Hardware setup for the test scenario with real-time input and static sensors' position. The screen shows the response of the system. Red dot: heading direction; blue dots: optical flow.

the results of a real-time test where the sound source was placed at 0° and some OF was generated by moving the hand around the boundaries of the eDVS.

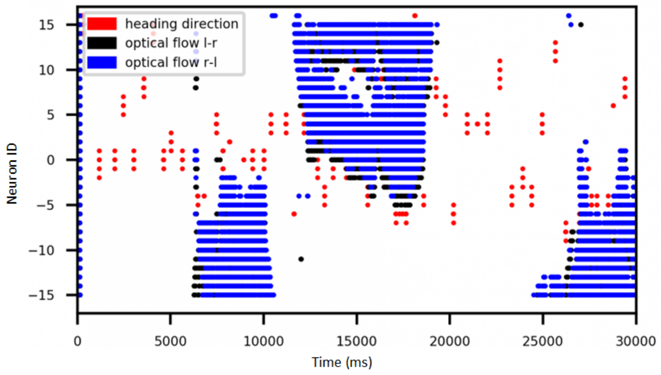


FIGURE 5.19: Output spikes from both the heading direction network and optical flow estimation network for the static hardware setup real-time test.

These findings were further investigated on a closed-loop robotic platform (Schoepe et al., 2020). A pan-tilt-unit was selected as robotic platform, where the event-based sensors were mounted as shown in Fig. 5.20. The ears-like microphone was place on the top of the platform, which was aligned with the center of the pan-tilt-unit's base motor. Then, the Dynamic Vision Sensor (DVS)

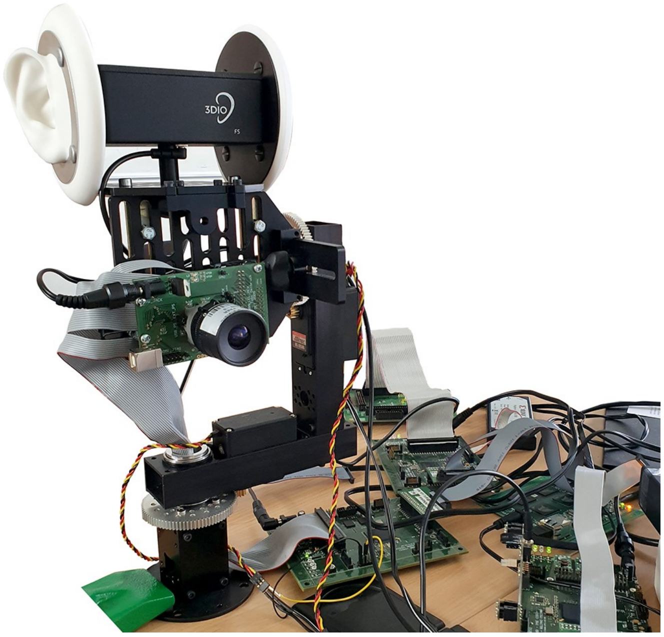


FIGURE 5.20: Hardware setup for the test scenario with real-time input and dynamic sensors' position. It is composed by the Neuromorphic Pan-Tilt-Unit including the Event-Based Camera, Neuromorphic Auditory Sensor with 3DIO Binaural Microphone, and SpiNNaker board.

was also aligned with the center of the microphone. However, it was placed on the middle of the pan-tilt-unit due to space restrictions.

The microphone was connected to one AER-Node FPGA board configured with the same NAS architecture detailed in Section 5.2.2.3. This board was directly interfaced to the SpiNNaker board through the SpiNN-link interface by using a custom PCB. The motor was also connected to this board in order to be controlled by the events coming from the head direction network on SpiNNaker. In a similar way, the DVS camera was connected to another AER-Node FPGA board to collect the output events and send them to the SpiNNaker board by using the second SpiNN-link interface available on the SpiNN-3 machine.

Regarding to the SNNs models, the network's architecture kept unchanged. The two networks (optical flow encoder network and a sound source direction network) feed into the sensory integration network which chooses the system's heading direction. This heading direction information controls the position of a pan-tilt unit through a head direction cell network, as proposed in (Massoud and Horiuchi, 2010).

For testing the system, the sound source position was moved around different positions while the obstacles were simulated by waving their hands or moving an object in front of the retina. As it can be watched in this video [6], the system reacted to these stimuli in real time, being able to find the closest direction to the sound source while avoiding obstacles, and the pan-tilt-unit moved accordingly to the heading direction calculated in the SNN.

This movement is relatively slow compared to other real-time robotic applications (Falanga et al., 2019). However, the reason resides in the large integration time for the audio-visual information. A larger integration period means more accuracy on the localization or heading direction but a slower system. The auditory information can be processed in the microseconds range, while the visual information is processed in the milliseconds range.

This means that when only auditory information is integrated, the system could work faster. Therefore, one possible solution to speed up the system could be to decrease the integration time of the obstacle avoidance network, thus having less visual accuracy but more movement speed. Further research will be carried out on this aspect by studying the attentional mechanisms. According to (McDonald et al., 2000), the sound source localization could improve the visual perception in such a way that we could obtain faster but less accurate reactions to a new events (involuntary reaction) and also slower but more accurate reactions after knowing the new situation (voluntary reaction). Those findings would allow the implementation of the proposed audio-visual integration system into mobile robots, such as hexapods or 4-wheel robots, for performing neuromorphic autonomous navigation.

5.2.3 Neuromorphic implementation of auditory perception in the iCub robotic platform

From the beginning of the technological era, the existence of humanoid robots was a common desire for many researchers. Soon, this dream will come true, and humanoid robots will be a common presence among humans. These kinds of robots are now used for multiple tasks: object and sound recognition, visual scene understanding, human detection and recognition, speech understanding, grasping, manipulation and locomotion capabilities, and even robot-human interaction. However, these capabilities come at a considerable computational cost and often are deployed on remote computing systems, with the need of constant data transfer.

In the last years, emergent research fields, such as neuromorphic engineering, aim to look into biology, not only to take inspiration about how to design robots, but also to mimic how humans figure out both simple and complex tasks in order to obtain low-power embedded solutions for solving those tasks. A well-known example of a humanoid robot is iCub (Natale

[6] `https://youtu.be/MOr2moBMmEo`

et al., 2017). iCub is a humanoid platform that was designed and developed to be used as a testbed for algorithms and theories modeling aspects of human cognition, including learning, perception and motor control (Natale et al., 2021). It provides to researchers with a ready-to-use, complete, humanoid platform, with sophisticated kinematics, a human-like sensory system, and a mature documentation and software Application Programming Interface (API).

Although there is a wide list of works about audio signal processing, most of them are specialized in one task, such as specific feature extraction for audio classification, sound source localization, or keyword spotting. Nevertheless, a combination of all of those elements is desired to obtain the whole information provided by the auditory scene. This way, bio-inspired models of the Auditory Ascending Pathway (AAP), as the one proposed by (Dávila-Chacón et al., 2018), can be used as inspiration for the neuromorphic approaches of the hearing sense.

FIGURE 5.21: The iCub robot. Picture taken from (Parmiggiani et al., 2012).

In addition, many works can be found in the literature that show how iCub can learn through visual stimuli (Monforte et al., 2020; Iacono et al., 2018; Wiesmann et al., 2012; Pasquale et al., 2015), but only a few of them use audio-visual information (Tikhanoff et al., 2010; Gonzalez-Billandon et al., 2020). The combination of the visual information together with auditory information could improve the accuracy in the learning procedures, thus allowing the system to be more robust. Moreover, other complex problems, as the cocktail party problem

(Thakur et al., 2015) or any other attentional problem (Hambrook et al., 2017), could be further investigated from the neuromorphic engineering point of view.

In this section, the integration of the NAS, combined with the Superior Olivary Complex (SOC) model detailed in Section 4.2 (also called Neuromorphic Auditory Complex (NAC)), within the iCub platform is presented (Gutierrez-Galan et al., 2022). The main aim of this integration is to provide a neuromorphic alternative to the traditional methods for audio processing in order to reduce latency, power consumption, and computational cost. Furthermore, to the best of our knowledge, this would be the first time that a digital neuromorphic cochlea model has been integrated within a humanoid robot.

This integration has been carried out by implementing the model separately into an FPGA and the SpiNNaker platform (Painkras et al., 2013). On the one hand, the FPGA offers reconfigurability and real-time signal processing of the proposed neuromorphic model, as well as a direct interface to the iCub main controller. On the other hand, the SpiNNaker platform offers the environment for deploying very large SNNs that can simulate complex brain structures, such as the auditory cortex. In addition, the bidirectional communication between the iCub and the SpiNNaker offers a potential close-loop system for real-time, audio-visual neurorobotic applications.

5.2.3.1 The iCub robot

The iCub robotic platform allows researchers to work in any field related to robot design, development, fabrication, and programming, from hardware to software. On the one hand, research on new motor design, PCBs, new materials, and new ASICs are needed to build an efficient and versatile humanoid robot capable to be used in multiple applications. On the other hand, a high- and low-level software development is needed to control all the sensors and manage the input and output data produced by them.

Regarding to the hardware, iCub is provided with several input sensors: cameras as eyes, microphones as ears, and an electronic skin for providing the touch sense distributed around the whole body (including the fingers). As actuators, it has a speaker for talking, motors as muscles, and joints (Natale et al., 2021). The level of detail of the iCub design is so high that even it has two motors per eye for implementing the microsaccade movement.

A set of custom PCB are used for communicating with the sensors and motors. These boards configure the sensors and collect the data, which is then sent to the host for being visualized and processed. In the same way, the motors' controller boards read the motors' status, send the information to the host, and move the motors according to the user commands. Although the boards are distributed all over the robot, the main ones are located in the iCub's head, as shown in Fig. 5.22. It has mounted a Xilinx Zynq-700 System on a Chip (SoC) that manage the cameras, the microphones, and the skin. This FPGA board, as well as the iCub's central CPU, is communicated with the host computer, where YARP is executed for giving software support to iCub.

YARP (Metta et al., 2006) is an open-source middleware upon which the iCub low-level and application-level modules are developed (Glover et al., 2018). The goal of YARP is to minimize research-level development and collaboration by promoting code reuse and modularization, thereby minimizing the workload of infrastructure-level software development (Metta et al., 2006). It also allows to use the iCub robot from a high level of abstraction, thus avoiding the need for deep knowledge about robotics to implement real-time applications.

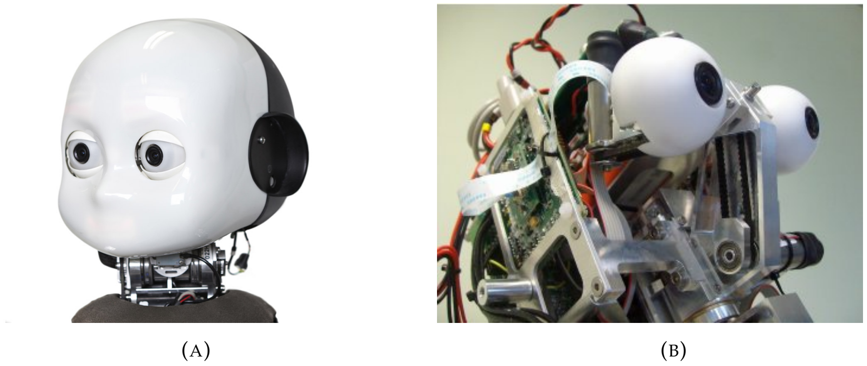


FIGURE 5.22: (A) Photograph of the iCub head electronics. (B) Photograph of the iCub head electronics. Pictures taken from (Parmiggiani et al., 2012).

Alternatively to the traditional version of the iCub, there is a neuromorphic version of the iCub robot which makes use of neuromorphic sensors and, therefore, neuromorphic computing and algorithms (Bartolozzi et al., 2011). The idea of this version is to investigate the advantages of the event-based representation of the sensed data, such as low latency, low power consumption, high parallelism, and data transmitting and storing optimizations. In addition, the asynchronous communication, inherent to event-based systems, reduces the latency by several orders of magnitude, thus allowing the implementation of fast sensorimotor closed-loop applications (as shown before), turning iCub as a perfect candidate to evaluate event-based, real-time robotic applications.

This neuromorphic version of iCub has two Asynchronous Time-based Image Sensor (ATIS) event-based cameras (Posch et al., 2010) as eyes instead of frame-based cameras. The skin data is also converted to events in order to have a neuromorphic representation of the touch sense. However, neither a neuromorphic cochlea model nor audio-to-events converter were included to replace the traditional audio processing models and algorithms. The software part was also updated by adding a new library that incorporates an event-based framework into the YARP middleware, thus allowing event-based sensors to be used within the iCub robotic platform (Glover et al., 2018). In this library, a multi-threaded event structure is provided to decouple the process of reading events into a data structure from that of running the algorithm. Modules are constructed such that the entire history of events is accounted for, but the processing algorithm runs only at the rate at which it maintains real-time operation.

According to (Glover et al., 2018), on the iCub robot, a Linux driver reads the events from the event-based sensors and the *zynqGrabber* module exposes the data on a YARP port. A packet of events is sent in a *ev::vBottle* (a specialized type of *yarp::os::Bottle*) such that the bit-coding of the AER is preserved: to retain data-compression and compatibility with other AER-based hardware. A module that receives a *ev::vBottle* can decode the AER and instantiate a *ev::vEvent* easily, as event decoding is provided by each event class. Encoding/decoding typically involves bit-shifts and a typecast to interpret a specific range of bits as the correct data type. The decoded events are stored in a *ev::vQueue* which wraps a *std::deque<event<vEvent»*. The procedure to obtain the event-stream is, therefore, transparent to the processing module. Reading *ev::vBottle* from a port is typically done using callback functionality (i.e., only where data is present) as the event-stream is asynchronous. Events can be saved and loaded from a file using the standard tools in YARP as an event-packet is fully interpretable as a standard *yarp::os::Bottle*. Therefore, it is easy to save a dataset using the *yarpdatadumper* and replay it using the *yarpdataplayer*. This is done externally to the event-driven library, simply by connecting the event-stream to/from the aforementioned modules using YARP connections.

5.2.3.2 Hardware integration of the NAC

As it was mentioned in the previous section, the iCub robot integrates a set of boards inside its head. Among them, it has a Xilinx Zynq FPGA board, as shown in Fig. 5.23. This board, in blue color and highlighted with the orange circle, has multiple purposes. Firstly, it physically interfaces the two ATIS cameras, the skin, the pair of microphones, and the Ethernet connector for the network communication through a custom base board PCB. Fig. 5.23 shows the location of both ATIS cameras (pink) and also the left I2S microphone (yellow).

Secondly, all the events from the neuromorphic sensors, such as the ATIS and the skin, as well as the sound signals from the microphones, are collected by the HPU IP core module, that is deployed into the FPGA among other modules. This IP core not only collect spikes from the sensors, but also send and receives spikes to and from the SpiNNaker platform (Painkras et al., 2013), as shown in Fig. 5.23. In addition, the HPU core interfaces to the robot computational system through the Zynq CPU where a dedicated module manages the communication of spikes to the robot's middelware YARP and the event-based processing libraries (Glover et al., 2018).

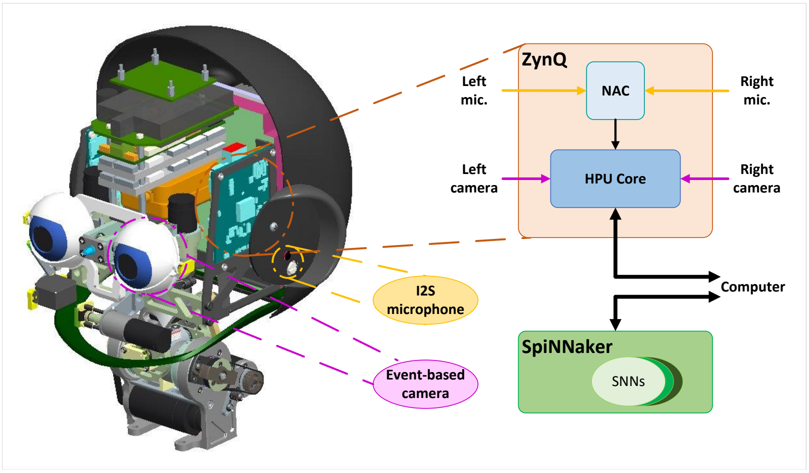


FIGURE 5.23: CAD view of the iCub head components with a block diagram of the proposed NAC-iCub integration. CAD picture taken from (Parmiggiani et al., 2012).

Thirdly, the idea was to integrate an auditory system model that comprises the NAC implemented on the Zynq FPGA, for reconfigurability and real-time signal processing, and the IC on SpiNNaker, where larger brain areas can be implemented. Thanks to the bidirectional communication between iCub and SpiNNaker through YARP, real-time, close-loop audio applications could be implemented.

For this first part of the integration, both the hardware and the VHDL modules were analyzed in order to select the best strategy to carry out the integration. The availability of free communication interfaces in the HPU core was checked. Once it was verified, the AER sensors map was reviewed, and a new sensor was proposed to be included. Then, several NAC models were generated and synthesized for obtaining the FPGA's utilization report and determine whether any of the proposed models would fit on it.

The NAC module was integrated within the iCub robotic platform as an IP core connected to the HPU module. It was finally composed by these 4 components (shown in Fig. 5.24):

1. A a 32-channels stereo NAS (Jimenez-Fernandez et al., 2017), that implements a spike-based cochlea model, converts the input audio signal to spikes, which are decomposed into frequency bands;
2. The SOC, detailed in Section 4.2, that implements the MSO model by using four frequency channels and sixteen neurons per channel;

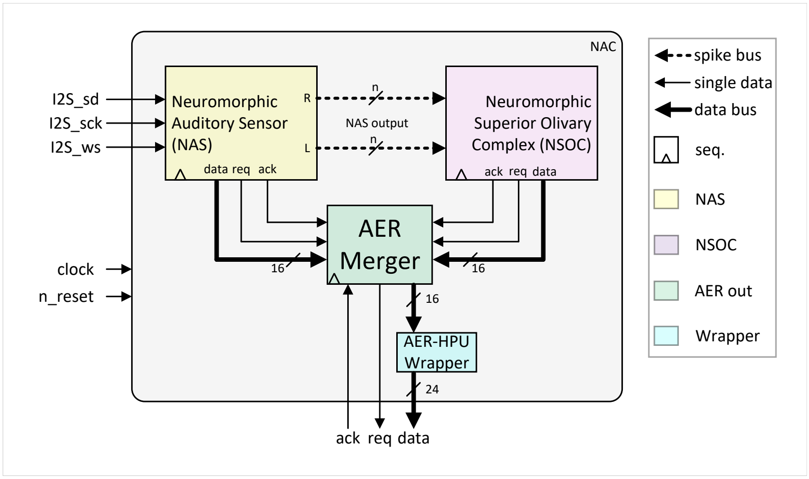


FIGURE 5.24: Block diagram of the NAC model integrated within the iCub robot. It is based on the one shown in Fig. 4.21, adding just the AER-HPU wrapper module.

3. The AER merger, that collects events from both aforementioned modules and interfaces the NAC to the HPU core by using the AER protocol through a dedicated port.
4. The AER-HPU wrapper, that adapts the NAC's output event package format (shown in Fig. 4.22) to the HPU's event package format (shown in Fig. 5.25).

Tables 5.5 and 5.6 summarize both the NAS and the SOC features in detail, respectively. Statistics provided by the synthesis report showed that the entire design consumed the 83.40% of the available LUTs, distributed in a 52.18% for the NAC (38.95% for the NAS and 13.21% for the SOC), a 10.41% for the HPU, and a 20.07% for other modules. Analyzing these results, it can be seen that the NAS module was the one that required most resources, even reaching the 100% of utilization when trying a 64-channels NAS. In addition, even though there are available resources for implementing a more complex SOC model, those resources were reserved for future improvements of the existing system.

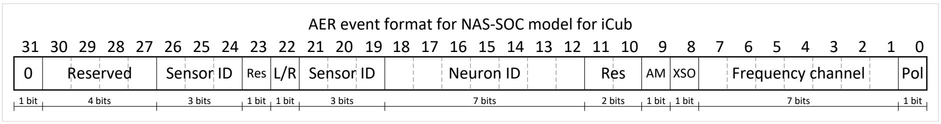


FIGURE 5.25: AER-HPU wrapper output event package format.

The event's package format matches with the AER sensors map, which was created to identify every part of the robot suitable to produce events in an organised way. Two different types of fields can be identified: 1) sensor identification, like *"Sensor ID"*, that is a unique code for each sensor, and *"L/R"*, which is only useful for stereo sensor architecture (eyes, ears, hands, etc), and identifies if the event is coming from the left or right sensor; 2) sensor's event information, like *"Neuron ID"*, that identifies the neuron of the SOC model which produced the event, *"AM"* (Auditory Model), that identifies if the event was produced by either the NAS or the SOC, *"XSO"*, that identifies if the produced SOC event came from the MSO or LSO models, *"Frequency channel"*, that indicates the NAS' frequency channel associated to the event for both NAS and SOC events, and *"Pol"*, that is the polarity of the event.

Comparing this format with the format shown in Fig. 4.22, some differences can be appreciated. The HPU format contains more fields, which are needed to process the event within the YARP framework. Furthermore, the position of some files have been changed in order to maintain the compatibility with the generic event's software class in YARP. Nevertheless, the HPU package is produced by the AER-HPU wrapper module, being thus independent from the NAC module and easily adaptable to new changes.

TABLE 5.5: Features of the NAS integrated within the iCub robot

	Parameter	**Value**	**Notes**
NAS common settings	NAS chip	Other	Zynq 7000
	NAS type	Stereo	Polarity merged
	Num. of channels	32	
	Clock freq. (MHz)	100	
Input interface	Audio input	I2S	All the values were by default.
Processing architecture	NAS architecture	Cascade	
	Start frequency	20 Hz	
	End frequency	22000 Hz	
	SLPF output att.	-36 dB	
Output interface	Spikes output	AER monitor	With the AER-HPU wrapper.

About the microphones, the standard pair of microphones placed in the normal iCub was replaced by a pair of MEMS microphones with I2S interface. These microphones, the ICS-43434 from InvenSense, have a wide frequency response from 60 Hz to 20 kHz, and a maximum sampling rate of 51.6 kHz in the high performance mode. Audio samples are collected by a HDL module that implements the I2S protocol as master. Then, this module sends the data to the Zynq CPU using a custom driver, which allows to save the audio samples into a .wav file by using a software application. Since the NAS in the NAC module already had implemented the I2S interface as input, also in slave mode, it

TABLE 5.6: MSO parameters.

	Parameter	Value	Notes
MSO	Num. of frequency channels	4	
	Start frequency channel	16	592.47 Hz
	End frequency channel	13	1166.80 Hz
	Num. of neurons per freq. channel	16	
	Detection time per channel	700	In microseconds
	Overlaping time	10	In microseconds

was connected to the microphones HDL control module, thus reading the audio samples and using them as input. This way, recordings in both .wav and event-based formats can be created using the same audio information for comparison purposes.

Finally, the sound features extracted with the NAC were pretended to be combined and integrated by means of more complex structures in order to obtain relevant information. For this purpose, multilayer SNNs were implemented on SpiNNaker, that model the functionality of the IC (Dávila-Chacón et al., 2018) and some parts of the auditory cortex (Dominguez-Morales et al., 2016). As shown in Fig. 5.23, the output of the NAC is collected by the HPU core and then sent to SpiNNaker in real-time through another dedicated bidirectional port, thus allowing the implementation of closed-loop applications.

5.2.3.3 Software integration of the NAC

After verifying that the hardware integration was working by adding a debug module to the FPGA project and checking by hand every data field of the event package, the hardware integration part was concluded. Therefore, the software integration process was started. It consisted of the addition of software modules for supporting the cochlea events on the YARP event-driven library (Glover et al., 2018). Thanks to the modular design of the YARP middleware, the integration of the new modules became easier.

Starting from the iCub robot, the *zynqGrabber* module, that runs on the Zynq FPGA, sends the collected events to YARP through a dedicated port by using the Transmission Control Protocol (TCP) protocol. Those events are received by the *vPreProcess*, which tags each event with a unique label that identifies the sensor which produced the event. Therefore, for each event, the *vPreProcess* first read the *Sensor ID* field of the event package, shown in Fig. 5.25, and creates a generic event object with the proper label and the rest of the event information. Then, this object is ready to be sent to any other postprocessing module, as it could be the visualization module or the application module.

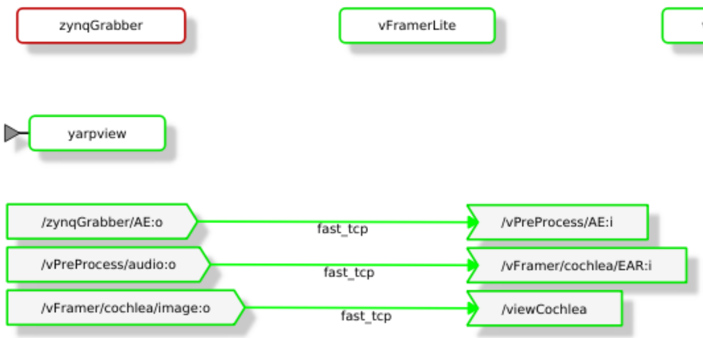


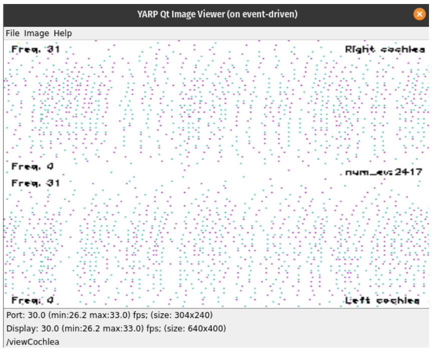
FIGURE 5.26: YARP modules diagram for the basic cochlea visualizer application shown in yarpmanager.

A new codec, called *codec_CochleaEvent*, was created to both code and decode the events generated by the NAC. In addition, a new port, called *audio:o*, was created in the *vPreProcess* module for sending out to other modules the events coming from the NAC. The word *"EAR"* was selected as label for fast identification of the NAC's events. A NAS real-time visualizer, shown in Fig. 5.27, was also implemented to check the cochlea's events activity.

The visualizer was split in two parts: 1) the bottom part, corresponding to the left cochlea, and 2) the top part, corresponding to the right cochlea. In addition, each part was also split in rows, from zero to thirty-one, where each row corresponds to different frequency bands in such a way that the bottom part represents high frequencies, starting at channel zero, and the top part represents low frequencies, ending at channel 31. Blue color means a positive event (polarity equal to zero), while magenta color means a negative event (polarity equal to one). It also shows the number of events per second.

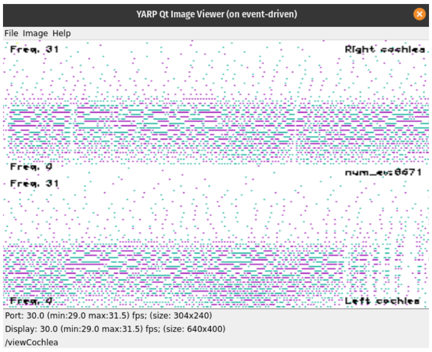
When no sound is played, both the background noise and iCub's fan noise are collected by the microphones, thus producing just noisy events, as shown in Fig. 5.27a. Nevertheless, these kinds of noisy events are considered as an intrinsic feature of the event-based sensors, and it could even help to make the system more robust (Fang et al., 2020). On the other hand, if a sound is played, the event's activity increases, as shown in Fig. 5.27b, and the visualizer shows the variation of the events' activity over time.

For collecting the NAC's events in a file, the *yarpdatadumper* software module was used. For each recording started by the user, a new text file with a predefined format was created. Each row contains the *vBottle*'s ID, the vBottle's timestamp, the event's tag (*"EAR"* for the NAC), and a list of event-timestamp pairs (provided by the *ZynqGrabber* module) grouped in the *vBottle* object. The event's format corresponds to the format shown in Fig. 5.25. Therefore, a



Port: 30.0 (min:26.2 max:33.0) fps; (size: 304x240)
Display: 30.0 (min:26.2 max:33.0) fps; (size: 640x400)
/viewCochlea

(A)



Port: 30.0 (min:29.0 max:31.5) fps; (size: 304x240)
Display: 30.0 (min:29.0 max:31.5) fps; (size: 640x400)
/viewCochlea

(B)

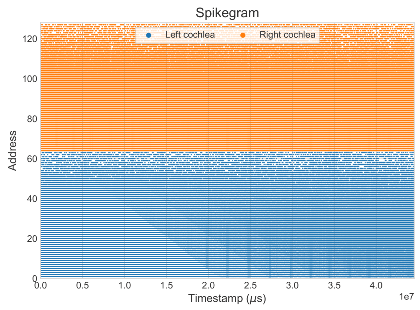
FIGURE 5.27: Cochlea visualizer window showing iCub's auditory events activity. (A) Events' activity when no sound is played. Those events correspond to the noise produced by the iCub's fan. (B) Events' activity when a 500 Hz tone is played.

decoding process is needed to extract the data of each package's field. A Python function was implemented to decode the *EAR* events and generate a text file containing the extracted information. This way, the file can be loaded into a postprocessing tool to analyze its content, like pyNAVIS (Dominguez-Morales et al., 2021b). This tool was updated as part of this thesis for supporting this format and being able to analyze the recordings.

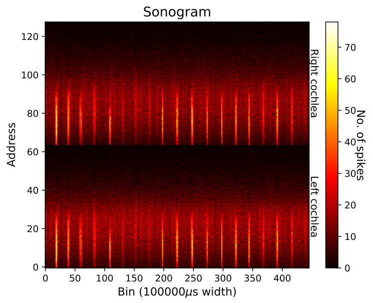
These recordings from *yarpdatadumper* can be loaded and played again by using the *yarpdataplayer* module. This feature allows researchers to repeat experiments using the same events as input stimuli, which is important for the learning procedure in complex tasks, like keyword spotting. A basic recording was collected as a proof-of-concept to analyze and validate the software part of the integration. Fig. 5.28 shows the output plots of a recording that consisted of a female placed in front of the iCub robot one meter away, and reading a list of keywords.

The spikegram of this recording, shown in Fig. 5.28a, does not allow us to see any useful information due to the high activity of the implemented NAS model. However, it can be visualized by generating both the sonogram and the average activity plots, shown in Fig. 5.28b and Fig. 5.28d, respectively. The sonogram shows us the frequency response of the input sounds for each frequency channel, giving an idea of the NAS' response to every single spoken word. This could be used to set the connection scheme properly according to those frequency responses.

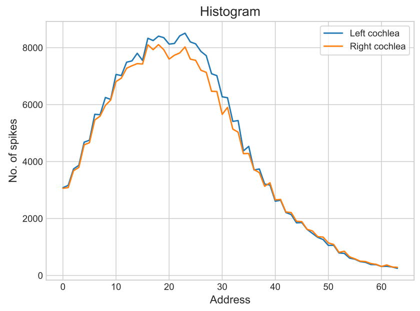
In addition, the average activity plot provides information about the number of spikes produced by each word, which could be useful for tuning the SNN associated with the keyword spotting detection. Finally, the histogram, presented



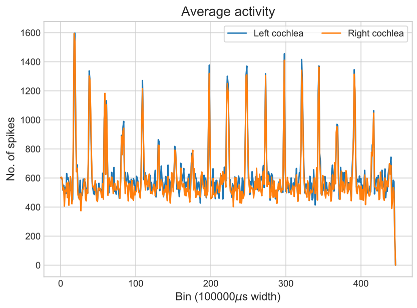
(A)



(B)



(C)



(D)

FIGURE 5.28: Example recording from a woman reading a list of words in front of the iCub robot, with a distance of 0.5 meters. (A) Spikegram. (B) Sonogram. (C) Histogram. (D) Average activity.

in Fig. 5.28c, shows that for a female voice, the main response is obtained in an address range between 14 and 30, which corresponds to a frequency range between 4500 Hz and 742 Hz. It is important to mention that the fan noise is mixed with spoken words, thus having a noisy dataset that could be improved by applying event-based filters before sending the data to SpiNNaker.

With this proof-of-concept, the software integration was successfully concluded. Therefore, the NAC-iCub full integration was considered as finished, thus being ready to be used for the researchers. The iCub's documentation was also updated by including the NAC features and the meaning of the NAC output, and the references related to the NAC module were included as well. At this point, the main task was changed to the development of auditory perception using the iCub robotic platform and the SpiNNaker machine.

5.2.3.4 Implementing an auditory perception application in real-time

Perception can be defined as the ability to interpret the information that we obtain through our different senses from the environment. This interpretation is an active process, and it depends on our cognitive processes and prior knowledge. Therefore, auditory perception could be defined as the ability to receive and interpret information that reaches the ears through audible frequency waves transmitted through the air or other means (Warren, 2013).

There is a series of processes to follow in order to perceive the sounds around us. Firstly, the information has to be received by the ear. When an object vibrates, which is the case of the human voice (vocal chords vibrate), the waves produced by this action are transmitted by the air or other means. When these waves reach the inner ear, certain cells are activated. Secondly, the information needs to be transmitted across the Auditory Ascending Pathway (AAP).

The signals produced by the cells are transmitted through different nuclei until it finally reaches the medial geniculate nucleus in the thalamus. Finally, the information is processed by the brain. The auditory information received by the ear is sent to the auditory cortex in the temporal lobes. The information is manipulated and sent to the rest of the brain to allow you to interact with it. Therefore, it can be said that the auditory perception is a multi-step process (Warren, 2013):

- Detection: if the sound has enough intensity to reach our ears.
- Discrimination: if we are able to differentiate the sound from other background noise.
- Identification: if we are able to identify where the sound is coming from.
- Recognition: if we recognize our personal relation with the sound (for example, "it's my friend's voice").
- Comprehension: if the sound is a message (someone telling us something), or the meaning of a sound (the bell showing that the class is over).

Auditory perception plays a crucial role in our day-to-day lives, being present in almost every task we perform. It allows us to properly interact with our environment, communicate with other people, alert us of any potential threats around us, and makes it possible to enjoy music. With this motivation, the implementation of an auditory perception model in the iCub robot was started. Taking the integration of the NAC as starting point, as well as the previous works of event-based sound recognition (Dominguez-Morales et al., 2016; Dominguez-Morales et al., 2017a; Dominguez-Morales et al., 2018a), the main goal of this implementation was to provide the iCub with the ability to listen a set of sounds, recognize them, create a map of the sound sources, and focus its attention on one of them by orientating the iCub's head towards the desired sound in real-time.

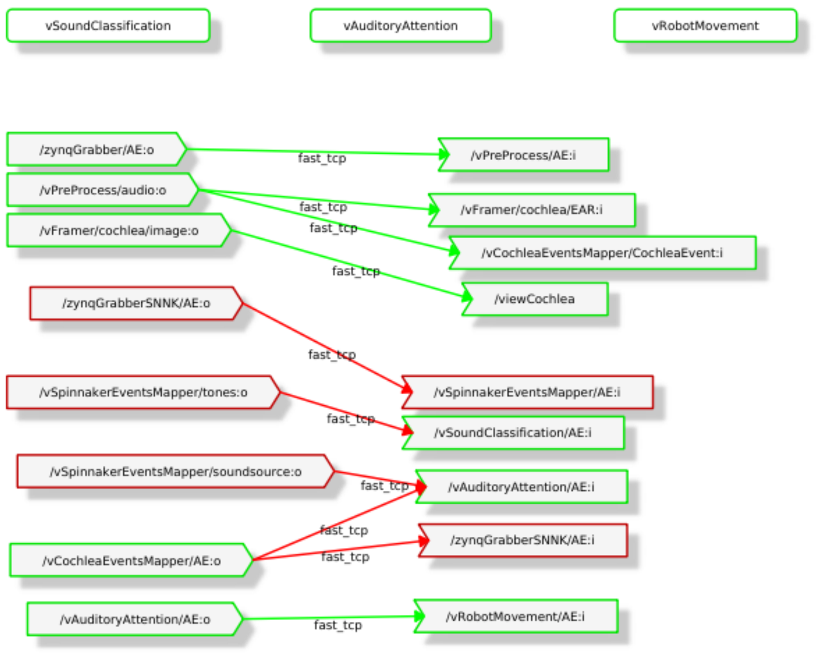


FIGURE 5.29: YARP modules diagram for the sound source localization application shown in yarpmanager.

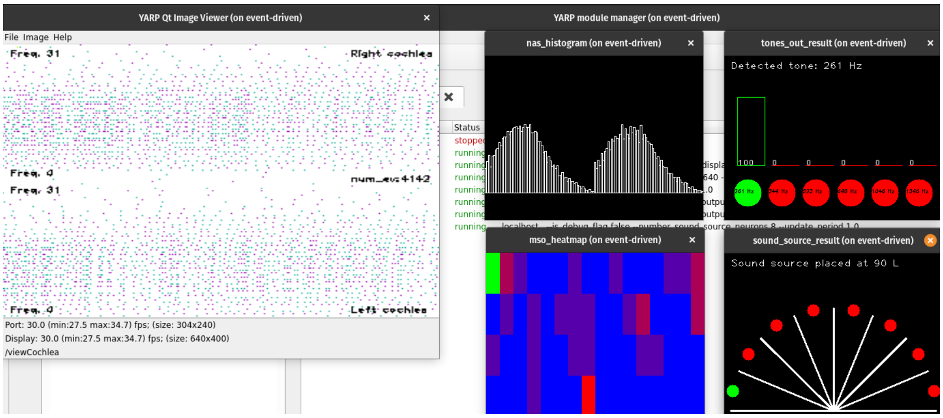
As it was mentioned in Section 5.2.3, part of the processing is carried out by the FPGA, and the rest is divided between the YARP framework and the SpiNNaker platform. Fig 5.29 modules that compose the auditory perception model. The modules involved on the events reception and visualization from the Zynq FPGA to YARP are the same as shown in 5.26. Then, the incoming events need to be mapped by assigning a unique address to each one for being later sent to either the SpiNNaker board or the next YARP processing module. This way, the decoding process is carried out only once, thus reducing the latency of the system.

This task is carried out by the *vCochleaEventsMapper* module, that takes *CochleaEvent*-type events as input and generates *AE*-type events as output. The addresses ranges are from 0 to 127 for the NAS and from 128 to 191 for the MSO since the NAC model architecture cannot be modified in real-time, and it is fixed as described in Tables 5.5 and 5.6.

In addition, this module also generates two more plots based on the NAC output events: 1) a visualizer for the MSO and 2) an alternative visualizer for the NAS frequency activity, shown in Fig. 5.30. For the first one, a real-time heat map plot was implemented due to its clarity for showing the main activity regions, thus allowing us to see if there is more than one sound source. For the second one, a real-time histogram visualizer was implemented to check the main frequency component of the input audio and to provide an idea of the nature of the input sound.

After the events mapping, two different options are available according to the hardware setup. If the SpiNNaker machine is not available, then the mapped events from the *vCochleaEventsMapper* are directly sent to the auditory attention module on YARP, called *vAuditoryAttention*. However, if the SpiNNaker machine is available, then the mapped events are sent to it through the module *zynqGrabberSNNK*.

In the second case, the events' addresses arrive to the external device population, also called vertex population, where the number of neurons is equal to the maximum number of addresses of the NAC: 192. Then, those events are properly sent to four different populations, as shown in Fig. 5.31, depending on the event's source. These four populations can be grouped in two groups: 1) sound recognition and 2) sound source localization.


YARP Qt Image Viewer (on event-driven)
File Image Help
Freq. 31
Right cochlea
Freq. 4
num_evs:4142
Freq. 31
Freq. 4
Left cochlea
YARP module manager (on event-driven)
Status
stopped
running
nas_histogram (on event-driven)
tones_out_result (on event-driven)
100
0
261 Hz
349 Hz
523 Hz
698 Hz
1046 Hz
1396 Hz
mso_heatmap (on event-driven)
sound_source_result (on event-driven)

```
Port: 30.0 (min:27.5 max:34.7) fps; (size: 304x240)
Display: 30.0 (min:27.5 max:34.7) fps; (size: 640x400)
/viewCochlea
```

```
Detected tone: 261 Hz
```

```
Sound source placed at 90 L
```

FIGURE 5.30: YARP visualizers for the auditory perception model.

For the sound recognition network, the multilayer SNN architecture presented by Dominguez-Morales et al. in (Dominguez-Morales et al., 2016) was used. This network was trained again with new recordings collected from the NAC in the iCub robot, and this training process was carried out automatically by implementing the functionality described in (Dominguez-Morales et al., 2016) in Python [7].

For the sound source localization part, a multilayer SNN was designed and implemented based on the architecture proposed by Davila et al. (Dávila-Chacón

[7]https://github.com/jpdominguez/Multilayer-SNN-for-audio-samples-classification-using-SpiNNake

et al., 2018). It is split in three different parts. The first part is composed by three networks which correspond to the excitatory output events coming from the MSO module in NAC, the excitatory output events coming from the LSO and the inhibitory output events coming also from the LSO module. In those networks, the neurons' activities are combined as shown in Fig. 5.32 left.

Only the MSO activity was used since the LSO model was not integrated within the NAC module yet. However, the network's architecture would be identical to the one used for the MSO network. Let is n the number of coincidence detector neurons in the ITD detection network. Let is m the number of neurons in the vertex population associated to the MSO model, being in this case 64 (16 coincidence detector neurons for each ITD detection networks, and 4 ITD networks connected from channel 13 to channel 16). Therefore, the connection scheme would be *(pre, post)*, where *pre* is equal to m and *post* is equal to m *mod(n)*. The weight of these connections are identical since the features extraction process is carried out in the next layer: the Inferior Colliculus (IC) layer.

The IC layer is composed by as many neurons as a single ITD detection network was set. In this case, the IC layer was composed by 16 neurons. The connection scheme was inspired by the one presented in (Dávila-Chacón et al., 2018). For each neuron x_i in the MSO layer of the 5.32, there is a projection to the neuron y_i of the IC layer with a weight w_i. In addition, for each neuron x_i in the MSO layer of the 5.32, there is also a projection to the neuron y_{i-1} and y_{i+1} of the IC layer with a weight w_j (except when i is equal to either 0 or $n-1$), where $w_i > w_j$.

This way, each neuron has not only local information but also the information of its neighbors, thus making the network more robust to noise and ambiguities. In fact, LSO neurons' activity would be also combined with the MSO neurons' activity in this layer to correct the ambiguities produced by the high frequencies sounds in the MSO nucleus. Moreover, the IC is able to adapt itself according to the input sound in both voluntary and involuntary ways (Schreiner and Winer, 2005; Lopez-Poveda, 2018), being this a key feature for auditory attention. However, this behavior was not implemented due to SpiNNaker does not allow to change the neurons' parameters in real time nor projections' weights and delays.

After the IC layer combine all the activity extracted from the auditory cues and generates and output which indicates the sound source localization, this information needs to be translated to a movement by taking the decision of how much the robot has to move its head and the direction of the movement. A Winner-Take-All (WTA) layer was implemented for taking the final movement decision. In addition, a reduction in the number of neurons was carried out in order to improve the decision accuracy (Dávila-Chacón et al., 2018). Fig. 5.32 left shows an example of the projections' scheme. Each neuron in the WTA layer, also called sound source population, receive inputs with high weight from two

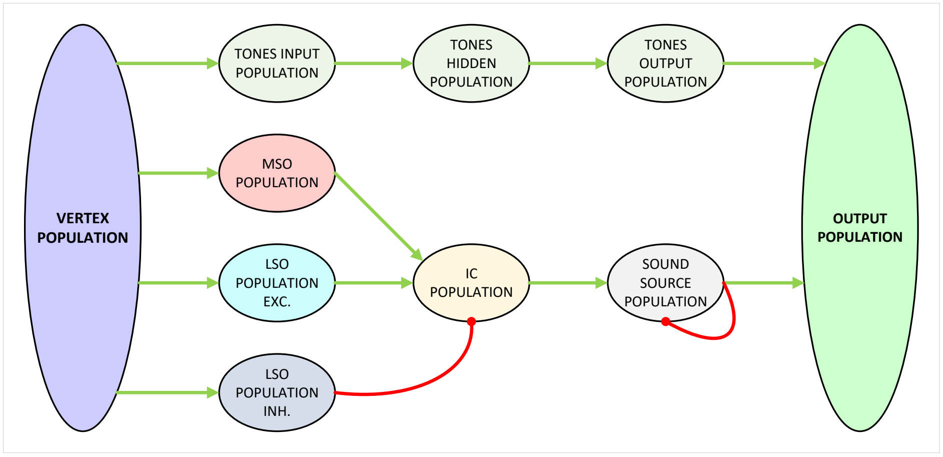


FIGURE 5.31: Overview diagram of the SNNs implemented on SpiNNaker to implement the auditory perception on iCub.

neurons from the IC layer, and also from the close neighbor of those two neurons, with lower weight in this case.

Finally, the outputs of both the sound source population and the tones output population are sent to an global output population with a one-to-one projection architecture. SpiNNaker allows bidirectional communication through the vertex population, which means that an external device can both send and receive events to and from SpiNNaker. Events coming from the external device are received on SpiNNaker by the vertex population, and then distributed to one or more networks. Therefore, the output events from SpiNNaker need to be grouped again in order to send them through the vertex population to the external device. Neurons' parameters in this output population are set in such a way that there exists a one-to-one event correlation, thus avoiding to loose information.

Taking again the Fig. 5.29 as reference, SpiNNaker output events are collected in YARP by the *zynqGrabberSNNK* module. Since events from both the sound recognition and sound source localization networks were mixed in the global output population, the *vSpiNNakerEventsMapper* module was implemented to split the events in two different sources according to their origin: 1) sound recognition and 2) sound source localization. Therefore, two post-processing modules were also implemented: 1) the *vSoundClassification*, and 2) the *vAuditoryAttention*.

On the one hand, the *vSoundClassification* module count the output events from the sound recognition network in a user-configurable time bin and indicates the winner neuron based on the events' activity, as it was done before in

(Dominguez-Morales et al., 2016). The result of this process is displayed through a normalized histogram in real-time, shown in Fig. 5.30 top right corner. By default, if there is not SpiNNaker activity, the neuron with ID equal to 0 (corresponding to 261 Hz pure tone frequency) is set as the winner neuron.

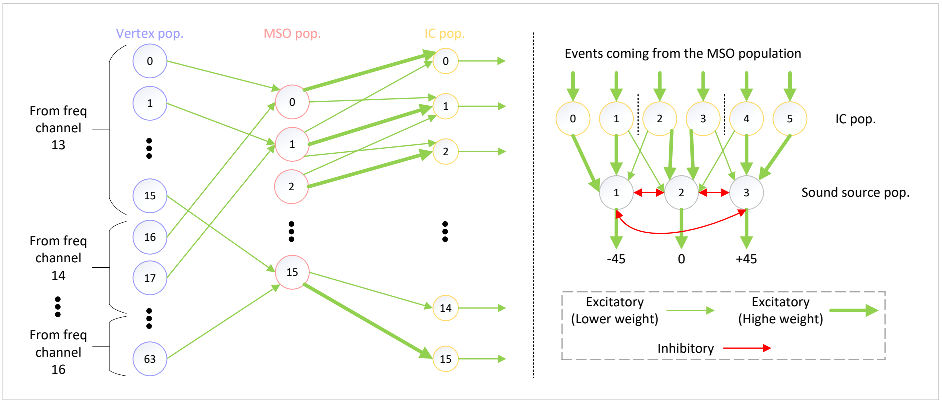


FIGURE 5.32: Spiking neural network diagram of the inferior colliculus model implemented on SpiNNaker.

On the other hand, the *vAuditoryAttention* module can take its input from either the *vSpiNNakerEventsMapper* module, if the SpiNNaker machine is available, or directly from the *vCochleaEventMapper* module, if the SpiNNaker machine is not available. In both cases, this module periodically integrates the input events to take the final decision about the position where the sound is coming from. The integration time can be tuned in real-time through YARP, and it is directly proportional to the desired detection accuracy in such a way that the higher the integration time is, the higher the accuracy will be. However, the integration time is inversely proportional to the movement speed. Therefore, a trade-off between accuracy and movement speed is needed, and it will depend on the input sound and the background noise.

Finally, the position estimation of the *vAuditoryAttention* module is received by the *vRobotMovement* module. This value indicates the absolute position where the iCub's head should be moved with respect to the origin. The iCub's neck movement range is [-60°, 60°], thus not covering the desired range of [-90°, 90°], being negative angles the middle-left region and positive angles the middle-right region of the horizontal plane. Therefore, the estimation was limited to that range, and it will be extended to the full range by moving also the torso along with the neck.

5.2.3.5 Datasets, tests, and preliminary results

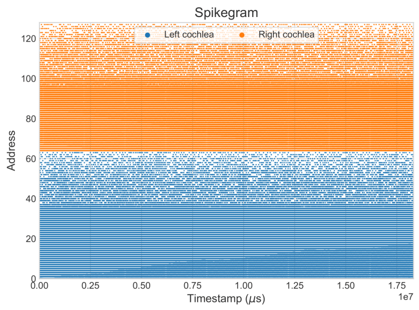
A dataset was created to both analyze and characterize the NAC's response within the iCub robotic platform under real working conditions. To this end, five different recording scenarios were established. First, the background sound of the laboratory where the robot was placed was recorded in order to have an estimation about how much noise was being added to the iCub's microphones. Four recordings of 20 seconds each were carried out while people were working, thus producing work-related sounds (keyboards, talking, walking, etc).

A continuous sound was detected after analyzing the recordings. However, it did not match any pattern of the laboratory sounds except the iCub's fans. The robot has one fan placed on the back part of its head, and two more fans in the backpack where the central computer is located. The noise produced by the head's fan inside the plastic cover is collected by the microphones, directly interfering with the sounds produced outside the plastic cover. New recordings were collected without any background noise (while the laboratory was empty) in order to analyze the sound features of the fan noise. The main frequency component was placed at 500 Hz, which was the same as the one used in Section 5.2.2.5 for the real-time tests. Although this problem can be solved, for example, by filtering the input sound or simply by replacing the fan by a more quiet version, it is out of the goals of this thesis, and it will be tackled in the future.

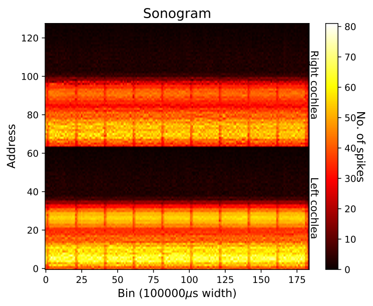
Second, six pure tones (261 Hz, 349 Hz, 523 Hz, 698 Hz, 1046 Hz, and 1396 Hz) were played at five different positions (left or -90^o, mid left or -45^o, middle or 0^o, mid right or 45^o, and right or 90^o) for 20 seconds each by using a speaker placed 1 meter away from the iCub robot and at the same horizontal plane of the iCub's microphones. The main purpose of these recordings were to train a SNN model similar to the one proposed by Dominguez-Morales et al. in (Dominguez-Morales et al., 2016) to perform real-time sound recognition. Thirty recordings were available to train the network by using either the method proposed in (Dominguez-Morales et al., 2016) or any other learning procedure. Differently from the original paper of pure tones recognition, where the recordings only contained sound information, these recordings also contain spatial information. Therefore, the sound recognition will be more robust since the system will be able to recognize the sounds even when they are not played just in front of the robot.

Fig. 5.33 shows the spikegram (A), sonogram (B), histogram (C), and average activity (D) plots obtained from a recording of the pure tones dataset, where a 523 Hz pure tone was played at the left side of the robot. The sonogram, as well as the histogram and the average activity, shows that the activity of the left cochlea is higher compared to the right one. In addition, the effect of the fan noise can be seen in the sonogram.

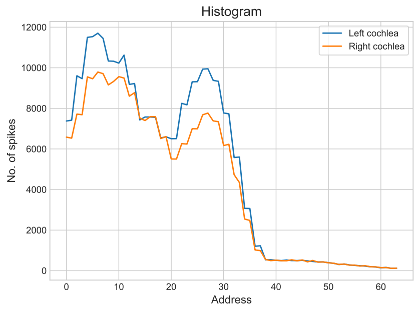
It also can be seen in the histogram that the frequency responses for both cochleas are identical. If we compare the number of total events of this recording to the one shown in Fig. 4.6c, the global event activity is lower in iCub (even



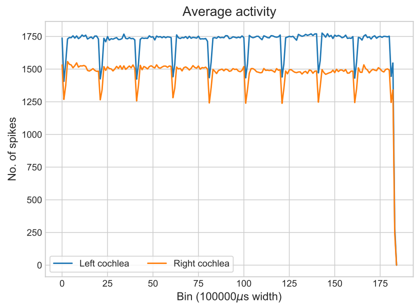
(A)



(B)



(C)



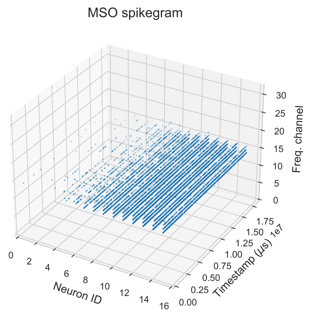
(D)

FIGURE 5.33: NAS response plots for a 523 Hz pure tone placed at the left. (A) Spikegram. (B) Sonogram. (C) Histogram. (D) Average activity.

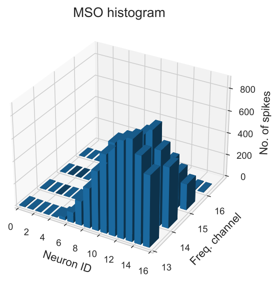
though the clock frequency of the FPGA was higher) due to both the head shape and the real-time test scenario, where no line-in connector was used. Furthermore, by simply checking the average activity values, the ILD estimation could be performed since the intensity difference is clearly visible. Note that the peaks observed in the average activity plot correspond to the repetition in loops of the sound track.

After analyzing the NAS' response, the MSO plots were generated. Fig. 5.34 shows the MSO spikegram (A), the MSO histogram (B), the MSO localization estimation (C), and the MSO heatmap (D) of the same recording used in Fig. 5.33. The spikegram shows that the main model activity is located at the higher neuron's IDs, which correspond to a big time difference between a sound arriving before to the left and then to the right ear. Almost no activity is found in the lower neuron's IDs.

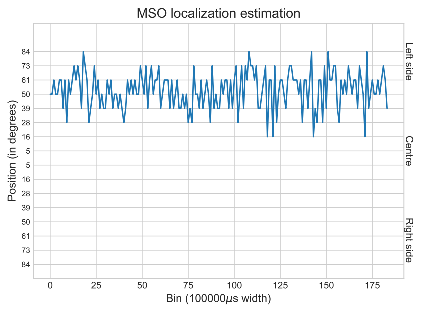
From the MSO histogram plot, it can be observed that the neuron with the highest activity corresponds to the neuron's ID 13, and this condition is also matched by all the frequency channels. In addition, the ITD detection network associated with the frequency channel that has the middle frequency more close to the input pure tone frequency produced more events compared to the others. The obtained response was distributed by following a pyramidal shape, thus matching with the expected responses according to the biological measurements (Bear et al., 2020).



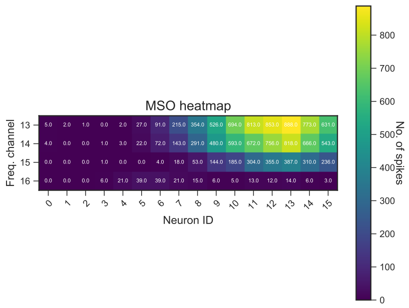
(A)



(B)



(C)



(D)

FIGURE 5.34: MSO response plots for a 523 Hz pure tone placed at the left. (A) MSO spikegram. (B) MSO histogram. (C) MSO localization estimation. (D) MSO heatmap.

A first estimation of the sound position, directly extracted from the raw MSO events, is shown in the MSO localization estimation plot. The result is a noisy signal approximately placed in the range $[40, 70]$ degrees. Although this result is not that accurate as desired, it can offer a first estimation of the sound position to later obtain a more accurate estimation by using SNN. There are some reasons which could explain this effect. Firstly, the absence of the ear shape surrounding the microphone in the iCub's head leads to a worst binaural cues extraction since the shape of the ear plays a key role for both the ITD and ILD.

Secondly, the fan noise, placed at the middle of the head, could be altering the ITDs by shifting the coincidences to the center neurons. This hypothesis could be tested by switching on and off the iCub's head fan and recording new samples under those conditions to later analyze them and check if the localization estimation was either improved or kept unchanged. However, the head fan cannot be stopped due to the thermal constraints of the chips. Therefore, new alternatives, like to create a virtual scenario simulating those conditions or to use a new iCub head with different boards, are needed in order to carry out these kinds of tests.

Thirdly, it may happen that the maximum time difference set for the ITD detection network for each channel does not match with the maximum ITD value associated to the iCub according to the distance between both microphones. This distance is 0.136 meters according to the iCub technical specifications (Beira et al., 2006). If we consider the speed of sound in air as $340m/s$, the time that the sound need to travel across the iCub's ears is $400\mu s$ when the sound is placed at 90 over the azimuth in one hear, i.e, when the ITD is maximum. However, the maximum ITD set for the iCub integrations was $700\mu s$, thus having $300\mu s$ of error taking also into account the reverberation.

If the number of coincidence detector neurons for each ITD detection network was set to 16 (See Table 5.6), then we can calculate, in a linear way, the time difference that each neuron can detect as follows:

$700\mu s/16 neurons = 43.75\mu s/neuron$

Therefore, the number of coincidence detector neurons needed to detect up to $400\mu s$ can be calculated as follows:

$400\mu s/43.75\mu s/neuron = 9.14 neurons$

Taking the ceiling of the result, it means that 10 neurons would be needed to have that maximum detection time of $400\mu s$. Then, the minimum resolution, in degrees, for each neuron can be calculated by dividing the detection range, set to 180º (from 90º at the left to 90º at the right, in front of the robot), by the number of coincidence detector neurons in the ITD detection network, set to 16. This way, we obtain the following:

$180°/16 neurons = 11.25\ °/neurons$

Meaning that each individual coincidence detector neuron has a detection range of 11.25°. If the needed number of neurons to detect up to $400\mu s$ was estimated as 10, it means that the absolute azimuth range of the iCub robotic platform can be easily estimated as follows:

$11.25°/neurons * 10 neurons = 112.5\ °$

The calculated detection range, 112.5 °, almost matches to the iCub's neck's movement range, which is set to 120 °according to the technical documentation[8], split on 60 °from the centre to the left side, and 60 °from the centre to the right side. This way, it can be said that the relative azimuth range which is able to be

[8]http://wiki.icub.org/wiki/Manual

detected by the iCub robot with the ITD detection network would be calculated as:

$112.5°/2 = 56.25°$

Thus having a region of 56.25°from the centre to the left side and another region of 56.25°from the centre to the right side. Two conclusions can be extracted from this result which would support this third hypothesis about the shift effect obtained in the results. Firstly, the maximum ITD value that can be extracted, and therefore the maximum detection angle in the azimuth, matches with the movement range of the iCub's neck joint, enabling to create a direct correlation between the estimated sound source in the space and the neck's joint position.

Secondly, the obtained value (56.25°) also matches with the mean value of the signal plotted in Fig. 5.34c and with the neuron with the maximum firing activity in Fig. 5.34d, (neuron with ID equal to 13) that corresponds to the fifth neuron of the left side taking the neuron with ID 8 as reference ($5_{th} neuron * 11.25°/neuron = 56.25°$). Therefore, this effect could be avoided by finely adjusting the maximum ITD detection time of the ITD detection networks implemented in the NAC module. It is important to mention that this effect was not found in the results showed in Chapter 4.2.6 due to the sounds were recorded in a virtual scenario with a hear-to-hear distance necessary to have a maximum ITD of at least 700 μs.

Additionally, two more effects can be observed in Fig. 5.34d. Neurons out of the iCub ITD detection range (theoretically, neurons with IDs 0, 1, 14, and 15) also had firing activity. In this case, it is biologically plausible that, when the sound is presented on one of the sides, neurons from the other side (IDs 0 and 1) also fire events since the ITD could be sometimes bigger than the maximum ITD detection time, thus producing some ambiguities, as shown in Chapter 4.2.6. However, for the iCub configuration, this effect is shifted from neurons 0 and 1 to neurons 14 and 15. In addition, neurons 14 and 15 also presented firing activity because the input sound was noisy, thus maybe inducing bigger ITD values in the original signal and, therefore, making them to fire. Nevertheless, neurons 14 and 15 should have and firing activity similar to neurons 12 and 13, respectively, according to (Kandel et al., 2000), but these firing activities are slightly lower due to the reason mentioned before, i.e., reverberation and aliasing.

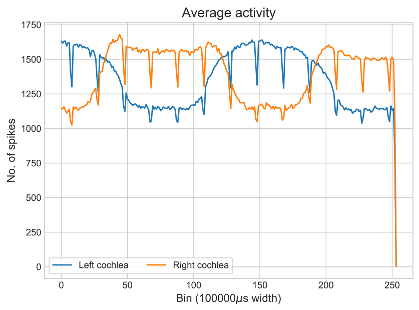
Finally, the global winner neuron belongs to the ITD detection network connected to the NAS' frequency channel number 16 (middle frequency set to 592.48 Hz), having fired 888 events. This result matches the expected behavior since the sound played was a pure tone of 523 Hz, which is close to the middle frequency of the band-pass filter. In addition, the winner neuron is the neuron 13 on each ITD detection network, thus demonstrating that the model is able to precisely extract the ITD values from the real-word input sounds. It can be also seen that the global model response has the shape of an inverted triangle due to the tuning parameters of the NAS' band-pass filter bank, where the maximum detection time is directly proportional not only to the ears' distance but also to the characteristic frequency of the band-pass filter. This way, the higher the middle frequency is, the lower the ITD maximum detection time should be, thus coinciding with the inverse of the middle frequency (i.e., the period). Although in this first version of the integration the maximum detection time was set with the same value for each network, in future versions this parameter will be set according to the input sound, thus turning this model to its adaptative version along with the adaptative version of the NAS.

After analyzing the data collected from static sound sources, a third set of recordings were carried out using dynamic sound sources. The iCub head was moved in a sinusoidal way from left to right and back (with a movement range of $[-60°, 60°]$), while the sound source was placed fix in the middle position, 1 meter away from the iCub robot. Therefore, from the point of view of the ears, we have a sound source moving from left to right. Then, the same six pure tones (261 Hz, 349 Hz, 523 Hz, 698 Hz, 1046 Hz, and 1396 Hz) were played for 25 seconds each. The main aim of this test was to characterize the model's response to a dynamic sound source, which could be a person talking in front of the iCub while walking from left to right. Furthermore, to check if a SNN model of the IC can improve the first estimation performed by the system without any MSO's output events post-processing, as it was already done in (Schoepe et al., 2020).

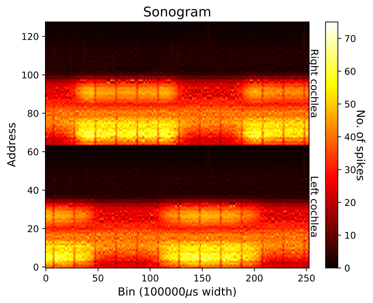
Fig. 5.35 shows the output plots from both the NAS model and the MSO model in response to a pure tone of 523 Hz played for 25 seconds. From the NAS plots, Fig. 5.35a shows the average activity of both left and right cochleae. It can be clearly seen the oscillatory movement of the iCub head, that was first looking at its right (left ear closer to the speaker) and then it starting to move its head towards its left (right ear closer to the speaker). This overall events' activity differences can be easily extracted by the LSO module to help the MSO in the sound source localization task. Although no time delays can be directly observed neither in Figs. 5.35a, 5.35b, 5.35c, Fig. 5.35e shows the output events of the MSO model. The extracted ITD values matches the movement of the robot's head, having a higher number of coincidences in the center neurons (since the probability of a coincidence is higher (Kandler et al., 2009)) compare to the extremes, where this activity is slightly lower.

For this experiment, the position estimation was calculated in pyNAVIS by accumulating events using a time bin and selecting the winner as the neuron that fired the most. The obtained result was a noisy position estimation, as shown in Fig. 5.35f. This approach, although acceptable for static sources, did not work properly for dynamics sources since it does not take into account the past. Therefore, each estimation is independent, leading the robot to carry out non-fluid movements. However, the final output can be improved by simply computing the accumulated mean of the estimated positions.

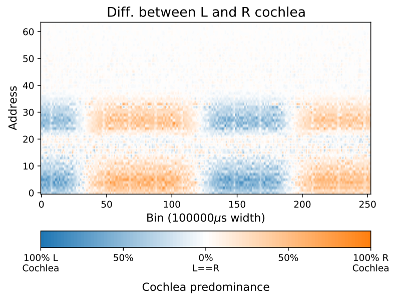
A bioinspired solution to take into account the temporal aspect of the output events was taken as an alternative to the traditional mathematical approach. A



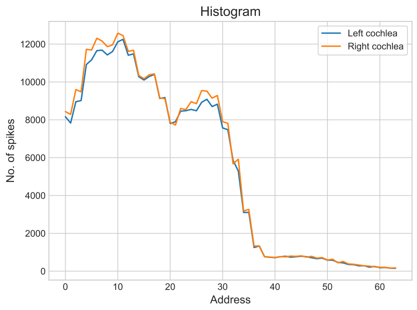
(A)



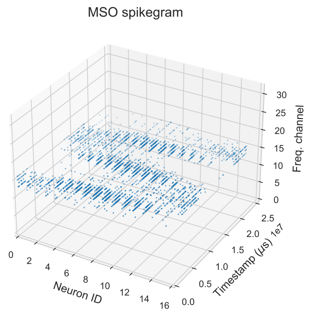
(B)



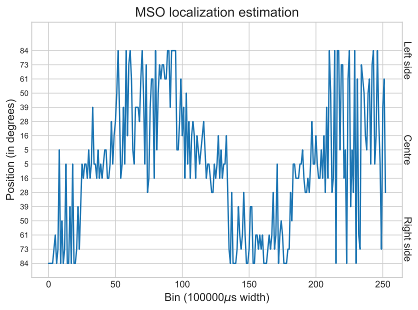
(C)



(D)



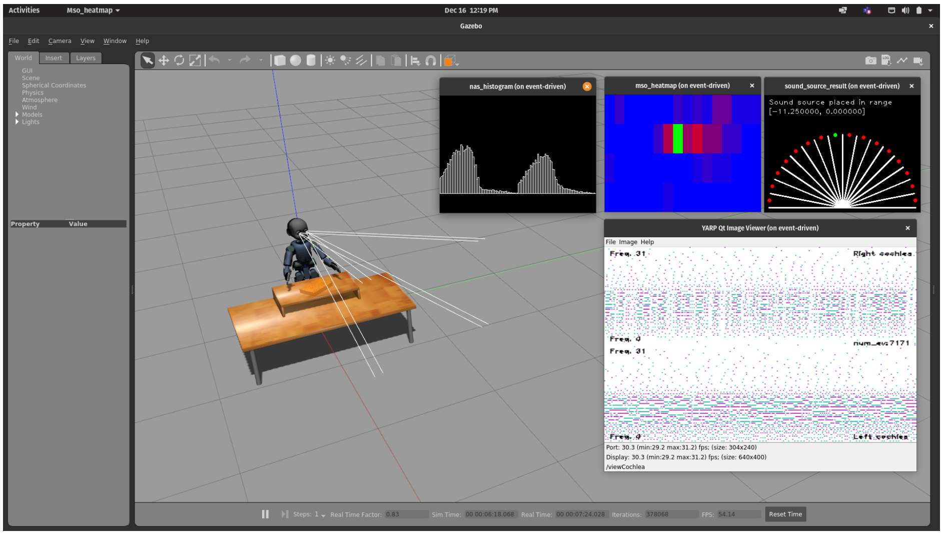
(E)



(F)

FIGURE 5.35: NAS' response plots (Figs. (A) Average activity, (B) Sonogram, (C) Difference, and (D) Histogram) and MSO response plots (Figs. (E) MSO spikegram and (F) MSO localization estimation) for a 523 Hz pure tone while moving the iCub head from left to right and back with a constant velocity.

model of the IC was designed and implemented by using SNNs to take into account not only the events but also the temporal relation between them, as introduced in Section 5.2.3.4.



Sound source placed in range
[-11.250000, 0.000000]

Port: 30.3 (min:29.2 max:31.2) fps; (size: 304x240)
Display: 30.3 (min:29.2 max:31.2) fps; (size: 640x400)
/viewCochlea

Real Time Factor: 0.83 Sim Time: 00 00:06:18.068 Real Time: 00 00:07:24.028 Iterations: 378068 FPS: 54.14

FIGURE 5.36: Screenshot of Gazebo simulator while performing a sound source localization simulation with a virtual neuromorphic iCub and real recordings as input stimuli.

A set of simulations were performed with the virtual neuromorphic iCub on Gazebo before using the real iCub robot. The recordings of dynamic sound sources were used as input stimuli by playing them using YARP data player. As main application, the sound source localization application in YARP (shown in Fig. 5.29) was used. Both options, with and without SpiNNaker, were tried in order to compare the output performance of both position estimation approaches. Fig. 5.36 shows a screenshot of Gazebo while running the simulation of the iCub robot moving its head towards the sound source, estimated by using the IC model on SpiNNaker, as well as the YARP visualizers for the NAC information.

It can be seen in the top right of the figure that the estimation carried out by the IC module matches the main activity shown in the heatmap visualizer, meaning that the model works according to the expectations. However, the iCub's head was not oriented to the front, which was the current position of the sound source at the moment of the screenshot. This is because the sound source was moving from left to right and the neck's motor needed some time to move from the left position to the middle position. The motor movement speed is an input parameter of the module and can be modified in real time.

Preliminary results of the sound source estimation performed by the IC model were obtained from the SpiNNaker simulation. Fig. 5.37 shows three raster plots that correspond to three of the entire auditory perception SNN model (See in Fig. 5.31). The top plot shows the output events from the four ITD extraction networks implemented in the NAC model that was integrated within iCub. Therefore, the same output pattern can be observed four times in the same plot. Bottom neurons belong to the ITD extraction network connected to the NAS' frequency channel number 13 and top neurons belong to the acITD extraction network connected to the NAS' frequency channel number 16 (See Table 5.6 for more details). The neurons' activity is higher for the neurons associated to the channel 16 since the input audio frequency was 500 Hz, which is close to the channel 16 middle frequency (592.47 Hz).

The second plot shows the output spikes of the IC SNN model, which combines the output from the ITD extraction network as explained in the Fig. 5.32. Similar to the localization estimation shown in Fig. 5.35f, the result is a stream of events organized in a sinusoidal shape. However, both extremes present a flat response (saturation response) instead of a pure sinusoidal shape. This effect is produced due to two main reasons: 1) the limitation of the ITD maximum detection time (explained in this section), and 2) the limitation in the movement of the iCub's neck ([-60°, 60°]) that does not allow to reach the entire 180 °range.

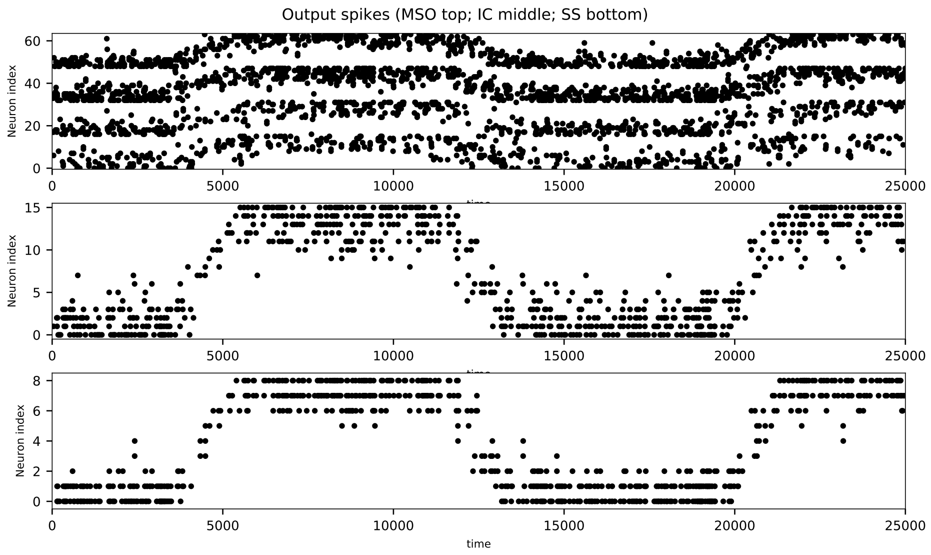


FIGURE 5.37: Output spikes for each gait pattern simulated on SpiNNaker.

Finally, the third plot shows the output spikes of the sound source localization network, which are sent back to YARP to and collected by the YARP module that moves the iCub's head. Although this plot looks less noisy compared to the middle one, it could be improved by training the network to obtain a more clear winner neuron. Some conclusions can be extracted from this plot. It seems the weights of the WTA's projections were not well set since more than one neuron were firing a high number of spikes at the same time. No learning procedures were used, and the weights were set manually following a trial-error procedure. A future improvement would be to study and apply a learning method to improve the WTA accuracy.

In addition, neurons' activity when iCub's head is facing to the sound source (i.e., the ITD is equal to zero) is lower compared to the activity when iCub is looking completely to either the left or right side. This could be produced by either a fast transition from one side to the other, and therefore not staying enough time in the middle position, or the fact that the head is stopped when it reaches the side, thus staying longer time at that position. It would be also interesting to evaluate the behavior of the network when more than one sound source (for example, one static source and one dynamic source) are used.

Beyond the pure tones dataset, two more datasets were generated. One of them corresponds to the spoken digits dataset, were a group of persons (both males and females) were seat one meter away in front of the iCub robot while counting from zero to nine, waiting two seconds between two consecutive numbers. A total of eight recordings were collected from eight different persons, four men and four women. A reference dataset can be found in the literature, the Heidelberg spiking data sets (Cramer et al., 2020). This dataset is composed by 10000 recordings of twelve different speakers in both German and English languages. The raw audios were converted to spikes by using a software version of a biological cochlea instead using a neuromorphic hardware-based sensor.

The other dataset corresponds to the keywords dataset, of which main purpose is to train a SNN that allows iCub to recognize and understand a set of speech commands and act according to them. As starting point, a set of eighteen words were selected: *"hi"*, *"ciao"*, *"iCub"*, *"robot"*, *"please"*, *"could"*, *"move"*, *"look"*, *"head"*, *"arm"*, *"right"*, *"left"*, *"up"*, *"down"*, *"fast"*, *"slow"*, *"start"*, *"stop"*. These words were considered as part of a most common sentence that a person could tell to iCub when interacting with it during a live demonstration. The recordings were collecting by following the same protocol as the spoken digits dataset, with a total of eight speakers (four women and for men), placed at one meter of distance with respect to the iCub's head, and waiting around 2 seconds between words.

To the best of our knowledge, there is not any other spike-based keywords dataset to be used as reference to compare the output spikes of the NAS model. In (Rasetto et al., 2021), authors converted part of the Google's Speech Commands Dataset (Warden, 2018) to spikes using a 32-channel monaural NAS model, generating 7592 AEDAT files. This dataset is composed by 30 short words spoken by thousands of different people (Rasetto et al., 2021) using different recording devices. However, the recordings of the presented spoken words dataset were recorded using directly a neuromorphic hardware without saving a traditional audio file before. Nevertheless, it would be possible to save also the raw audio samples as .wav files since the microphones' output data lines are connected to both the NAC and to the traditional sound processing module in the iCub's Zynq FPGA. The main advantage of having the input audio in both formats (.wav and events) will lead to a better comparisons between the traditional methods of sound processing and the new approaches given by the neuromorphic engineering.

These two datasets are being used to train a generic SNN model based on previous works which already used a NAS model to perform words recognition (Dominguez-Morales et al., 2018b; Rasetto et al., 2021). This task is still under development due to the complexity of the training task, not having yet enough conclusive results to mark this process as done. In the future, the main purpose will be to combine this network along with the sound source localization network to let the iCub orientate its head towards the person who is talking to it, and therefore start doing research in attentional mechanisms.

5.3 Is it worth to do the effort?

Most of the people, including researchers, would ask: *"Why is it necessary to develop this neuromorphic approach of the audio processing if it already exists a traditional alternative which works 100% good?"*. This question is full of small aspects that can be discussed from different points of view, and all of them could be right. In this section, the conlusions found during the design and implementation of these three cases of use are discussed.

Firstly, the technical difficulties that were found while interfacing all the components of the system. Normally, a simple demonstrator can be composed by up to four different boards, being this boards connected by multi-line cables and different connectors on each board. Taking into account that mobile robots were used, several failure points can be detected. In addition, different operational voltages may be required, thus being needed voltage translators to be able to communicate different devices. This kind of problems are common at any approach of audio processing since they are intrinsic to the real-time mobile robots. However, due to nature of the event-based information, neuromorphic robotics can be considered more robust to the noise in the cables or event to a connection failure due to their ability to adapt its behavior in real-time (Thor et al., 2021). Therefore, even from the technical point of view, we can say that it is worth to do the effort to use neuromorphic hardware to develop a new generation of robots.

Secondly, it is important to mention that in the last years, the algorithms to train SNNs have improved their accuracy and usability. However, still nowadays is difficult to train this kind of networks for a generic application. For example, to recognize two words by using traditional methods can be considered an easy task, but the same task is a hard problem when working with neuromorphic auditory sensors and SNNs (Rasetto et al., 2021). Then, why should we use these networks and this approach? Although right now there are not so many advantages, the noise tolerance and the adaptability are two of the most important features that the SNN have demonstrated to work. Maybe it is still too early, and the technology needs to improve a bit, but in some neuromorphic visual applications, it has been demonstrated that the neuromorphic approach beats the traditional approach.

Finally, the neuromorphic engineering can be considered as a young field of research, and a lot of work is still needed to achieve big results. The desired to develop more biological plausible devices which could interact directly with the brain is the biggest motivation of the researchers since it may help the people to recover the vision, the hearing, or the touch sense, thus improving their life qualities. Therefore, as conclusion of this thesis, we can say that it is totally worth to do the effort to think, design, develop and test every single idea that could lead to improve the state of the art about the hearing sense from the point of view of the neuromorphic engineering.

Chapter 6

Conclusions and future works

"Research is what I'm doing when I don't know what I'm doing"

– Wernher von Braun

This chapter presents the conclusions of the thesis, its main contributions, and future works. Before introducing the conclusions, it is important to highlight that the code generated during the realization of this thesis was shared as open-source projects. A list of repositories, containing both the repositories associated to the work of this thesis and the contributions to other repositories, is shown next:

-Own repositories:

- `https://github.com/RTC-research-group/OpenNAS`
- `https://github.com/dgutierrezATC/NASIC`
- `https://github.com/dgutierrezATC/nssoc`
- `https://github.com/dgutierrezATC/TDE_vhdl`
- `https://github.com/dgutierrezATC/NeuroPod`
- `https://github.com/event-driven-robotics/NAS_iCub`

-Contributions to external repositories:

- `https://github.com/dgutierrezATC/NAS_SpiNNaker_interface`
- `https://github.com/dgutierrezATC/GenericSeqMon`
- `https://github.com/dgutierrezATC/event-driven`
- `https://github.com/robotology/event-driven`
- `https://github.com/jpdominguez/pyNAVIS/tree/ssoc_integration`
- `Multilayer-SNN-for-audio-samples-classification-using-SpiNNaker`

6.1 Conclusions

In this section, the main contributions and conclusions, extracted from the work presented throughout this document, are highlighted:

- An in-depth study of the biological principles of the human auditory sense, as well as the most relevant models for mimicking it, has been carried out. In addition, a study of the biophysical properties of biological neurons has been conducted, along with an analysis of different codification strategies for spike-based signals.
- A state-of-the-art analysis about the use of bio-inspired auditory processing systems in robotics was conducted. Several common weaknesses were identified, being an inspiration for this work.
- Since a FPGA-based Neuromorphic Auditory Sensor (NAS) was developed by our research group, it was decided to make it open-source in order to allow the neuromorphic community to use it. Therefore, an open-source software tool, called OpenNAS, was developed to guide the user through the design, configuration, and deployment of a NAS architecture by means of a five-steps wizard. This work was published in the Neurocomputing journal, presented in many Neuromorphic Engineering workshops and conferences, and it is being currently used in many research centers, as the IMSE-CNM (Seville, Spain), IIT (Genova, Italy), CITEC (Bielefeld, Germany), among others. To the best of our knowledge, this is the first software tool that generates the VHDL code of a neuromorphic sensor.
- The process of ASIC design was studied. An ASIC of a tiny NAS model was designed, fabricated, and tested. The model consisted of eight-frequency bands, mono NAS with both I2S- and PDM-based microphones interface and without output events monitor. Even though the measured behavior did not match with the expected one, this first approach of a digital, fully event-based ASIC was considered by us like the beginning of a long way to achieve medical neuromorphic devices, as neuromorphic cochlear implants.
- Due to the importance of the sound source localization task and its applications, a digital, FPGA-based, event-based model of the Superior Olivary Complex (SOC) was designed, implemented, and tested. This design contains the Anteroventral Cochlear Nucleus (AVCN), the MSO, and it is ready to contain the LSO, which is currently under development. This model can be adapted according to the application's requirements, and it has been also integrated within the NAS in order to take advantage of the intrinsic parallelism offered by the FPGA fabric, thus avoiding bottleneck problems and, therefore, timing accuracy. This model has been used in multiple real-time robotic applications, and it was presented in the Embedded Systems Week Conference (ESWEEK) 2019.

- Since the most relevant information about the localization is encoded in the time domain, an alternative to the Jeffress model for the ITD encoding was proposed. The Time Difference Encoder (TDE) model was used as reference, and to the best of our knowledge, the first digital implementation of the TDE model for FPGAs was designed and implemented. The model was also validated with a proof-of-concept sound source lateralization application, proving the usability of this model for such kind of tasks. This work was published in the IEEE Transactions on Neural Networks and Learning Systems (IEEE TNNLS) journal 2021.
- An audio-guided SCPG was developed to control an hexapod robot in real time, called NeuroPod. In this work, carried out during the one-month research internship in the University of Cadiz, with the Prof. Fernando Perez-Peña, a 64-channels, mono NAS was used for classifying pure tones and to change the motors' positions according to the identified sound. The auditory information was sent in real-time from the FPGA to the SpiNNaker board, where three different SNNs were implemented. Finally, the movement spikes were sent back to the FPGA, where the motors are commanded. This work was published in the Neurocomputing journal and presented in the International Symposium on Circuits and Systems (ISCAS) 2019.
- A novel neuromorphic sensory integration architecture was designed as part of an international collaboration through several research internships in the Neuromorphic Behaving Systems Group of the University of Bielefeld, headed by Elisabetta Chicca. This architecture was composed by an event-based camera, the DVS, an event-based auditory sensor, the NAS, and the neuromorphic hardware SpiNNaker. The goal was to perform autonomous navigation of a moving robot that follows a sound source while avoiding obstacles. One of the main contributions was to combine audio-visual information in a neuromorphic hardware and to take decisions in a close-loop system. This work was presented in the Biomedical Circuits and Systems Conference (BioCAS) 2019 and in the International Symposium on Circuits and Systems (ISCAS) 2020.
- To the best of our knowledge, the integration of an event-based neuromorphic auditory system within the iCub robot was performed for the first time. This integration was carried out during the five-month research internship in the Event-Driven Perception for Robotics Group at the Istituto Italiano di Tecnologia, lead by Chiara Bartolozzi. Several adaptations were applied to both the hardware and software parts, as well as new functionalities were implemented for visualizing the auditory information. A demonstrator was implemented, where the SpiNNaker board was used to perform SNNs the sound recognition and localization. This work was presented in the Neuro Inspired Computational Elements Conference (NICE) 2022.

6.2 Future works

The use of the OpenNAS tool has increased significantly since it was presented as an open-source project to the community. Some installation and usage issues have been reported by other users related to operating system incompatibilities. Therefore, a cross-platform Python version of the OpenNAS tool would be desirable in order to open this software to a wider range of users, and, thus, to make it more accesible to the neuromorphic community and facilitate its use. In addition, the SOC model generation would be added to the tool as well for unifying the generation process.

Regarding the NASIC, the obtained results from the ASIC characterization did not match with the expectations. Many aspects from the ASIC design flow could be improved, starting for the VHDL files. Improving the design by using the feedback of the Computer-Aided Design (CAD) tools and applying good designing practices would prevent the next version of the NASIC from unexpected behaviors. In addition, further verifications and tests will be carried out, which were skipped due to the lack of experience. As a future work, a new ASIC will be designed, including debug features and external configuration.

Further work is also needed for the proposed model of the SOC. While the MSO part was already tested, the LSO part is still under development. Moreover, an important feature, like the real-time adaptation, is still not included. In the future, an adaptative version of the SOC model, along with an adaptative version of the NAS, will be implemented for performing unsupervised auditory attention tasks. Also, the study of the application of the TDE model in the sound lateralization task will continue. The performance in terms of accuracy will be analyzed, and a comparison with the state-of-the-art results will be performed in order to determine the advantages and disadvantages of this approach.

In the field of robotics, the combination of the work carried out for the audio-guided SCPG and the work performed in the development of the sensory integration network will result in a robust, autonomous, fully event-based, mobile system. By improving the accuracy in the navigation task and adding unsupervised learning procedures, this kind of robot could be used in rescue or risky tasks where the adaptation to unknown situations is critical.

Finally, the integration project of the NAS within the iCub robotic platform has to be finished. Although a functional version was successfully achieved, some aspects need to be improved. For example, the sound recognition network accuracy is lower than the original work, and the sound source localization task could achieve a higher accuracy if a better training strategy is followed. In the future, both visual and auditory event-based signals will be combined for solving audiovisual attention-based tasks in real-time, like speech separation during the cocktail party effect or object recognition through speech commands.

Chapter 7

Bibliography

"If you don't have time to read, you don't have the time (or the tools) to write. Simple as that."
– Stephen King

Alex, Laird (2009). "The Von Neumann Architecture Topic Paper #3". In: *Computer Sience* 319, pp. 360–8771.

Altshuler, Douglas L. and Mandyam V. Srinivasan (2018). "Comparison of Visually Guided Flight in Insects and Birds". In: *Frontiers in Neuroscience* 12, p. 157. ISSN: 1662-453X. DOI: 10.3389/fnins.2018.00157.

Amagai, S., C.E. Carr, and R.J. Dooling (1996). "Brainstem auditory time-coding nuclei in budgerigars: Physiology". In: *ARO Abstracts* 19, p. 191.

Ambroise, Matthieu, Timothée Levi, Sébastien Joucla, Blaise Yvert, and Sylvain Saïghi (2013). "Real-time biomimetic Central Pattern Generators in an FPGA for hybrid experiments". In: *Frontiers in Neuroscience* 7, p. 215. ISSN: 1662-453X. DOI: 10.3389/fnins.2013.00215. URL: https://www.frontiersin.org/article/10.3389/fnins.2013.00215.

Angelidis, Emmanouil, Emanuel Buchholz, Jonathan Patrick Arreguit O'Neil, Alexis Rougè, Terrence Stewart, Axel von Arnim, Alois Knoll, and Auke Ijspeert (2021). "A Spiking Central Pattern Generator for the control of a simulated lamprey robot running on SpiNNaker and Loihi neuromorphic boards". In: *arXiv preprint arXiv:2101.07001*.

Barlow, Horace B (1961). *Possible principles underlying the transformations of sensory messages*. MIT press.

Barron, Andrew B, Kevin N. Gurney, Lianne F. S. Meah, Eleni Vasilaki, and James A R Marshall (2015). "Decision-making and action selection in insects: inspiration from vertebrate-based theories". In: *Front. Behav. Neurosci.*

Barron-Zambrano, Jose Hugo, Cesar Torres-Huitzil, and Bernard Girau (2010). "Hardware implementation of a CPG-based locomotion control for quadruped robots". In: *International Conference on Artificial Neural Networks*. Springer, pp. 276–285.

Bartolozzi, C, P Ros, F Diotalevi, N Jamali, L Natale, M Crepaldi, and D Demarchi (2007). “Event-driven encoding of off-the-shelf tactile sensors for compression and latency optimisation for robotic skin.” In: *RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 166–173.

Bartolozzi, Chiara and Giacomo Indiveri (2007). “Synaptic dynamics in analog VLSI”. In: *Neural computation* 19.10, pp. 2581–2603.

Bartolozzi, Chiara, Giacomo Indiveri, and Elisa Donati (2022). “Embodied neuromorphic intelligence”. In: *Nature communications* 13.1, pp. 1–14.

Bartolozzi, Chiara, Francesco Rea, Charles Clercq, Daniel B Fasnacht, Giacomo Indiveri, Michael Hofstätter, and Giorgio Metta (2011). “Embedded neuromorphic vision for humanoid robots”. In: *CVPR 2011 workshops*. IEEE, pp. 129–135.

Bascuas, Luís Enrique López (1997). “La percepción del habla: problemas y restricciones computacionales”. In: *Anuario de psicología/The UB Journal of psychology* 72, pp. 3–22.

Bear, Mark, Barry Connors, and Michael A Paradiso (2020). *Neuroscience: Exploring the brain*. Jones & Bartlett Learning, LLC.

Beira, Ricardo, Manuel Lopes, Miguel Praça, José Santos-Victor, Alexandre Bernardino, Giorgio Metta, Francesco Becchi, and Roque Saltarén (2006). “Design of the robot-cub (icub) head”. In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, pp. 94–100.

Benesty, Jacob et al. (2009). “Pearson correlation coefficient”. In: *Noise reduction in speech processing*. Springer, pp. 1–4.

Berge, Hans Kristian Otnes and Philipp Hafliger (2007). “High-speed serial AER on FPGA”. In: *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*. IEEE, pp. 857–860.

Bhadkamkar, Neal and Boyd Fowler (1993). “A sound localization system based on biological analogy”. In: *IEEE International Conference on Neural Networks*. IEEE, pp. 1902–1907.

Boahen, Kwabena A (2000). “Point-to-point connectivity between neuromorphic chips using address events”. In: *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 47.5, pp. 416–434.

Braitenberg, Valentino and Almut Schüz (1998). “Cortical architectonics”. In: *Cortex: Statistics and Geometry of Neuronal Connectivity*. Springer, pp. 135–137.

Brette, Romain and Wulfram Gerstner (2005). “Adaptive exponential integrate-and-fire model as an effective description of neuronal activity”. In: *Journal of neurophysiology* 94.5, pp. 3637–3642.

Brosch, Tobias, Stephan Tschechne, and Heiko Neumann (2015). “On event-based optical flow detection”. In: *Frontiers in Neuroscience* 9, p. 137. ISSN: 1662-453X. DOI: 10.3389/fnins.2015.00137. URL: https://www.frontiersin.org/article/10.3389/fnins.2015.00137.

Büschges, Ansgar, Turgay Akay, Jens P Gabriel, and Joachim Schmidt (2008). "Organizing network action for locomotion: insights from studying insect walking". In: *Brain research reviews* 57.1, pp. 162–171.

Cant, Nell B and Christina G Benson (2003). "Parallel auditory pathways: projection patterns of the different neuronal populations in the dorsal and ventral cochlear nuclei". In: *Brain research bulletin* 60.5-6, pp. 457–474.

Cariani, P. (2011). "Jeffress model". In: *Scholarpedia* 6.7. revision #137337, p. 2920. DOI: 10.4249/scholarpedia.2920.

Cassidy, Andrew and Andreas G Andreou (2008). "Dynamical digital silicon neurons". In: *2008 IEEE Biomedical Circuits and Systems Conference*. IEEE, pp. 289–292.

Cerezuela-Escudero, Elena, Fernando Pérez-Peña, Rafael Paz-Vicente, Angel Jimenez-Fernandez, Gabriel Jimenez-Moreno, and Arturo Morgado-Estevez (2018). "Real-time neuro-inspired sound source localization and tracking architecture applied to a robotic platform". In: *Neurocomputing* 283, pp. 129 –139. ISSN: 0925-2312.

Cerezuela-Escudero, Elena et al. (2013). "Spikes monitors for FPGAs, an experimental comparative study". In: *International Work-Conference on Artificial Neural Networks*. Springer, Berlin, Heidelberg, pp. 179–188.

Chan, Vincent, Craig Jin, and André van Schaik (2012). "Neuromorphic Audio-Visual Sensor Fusion on a Sound-Localising Robot". In: *Frontiers in Neuroscience* 6, p. 21. ISSN: 1662-453X. DOI: 10.3389/fnins.2012.00021.

Chan, Vincent et al. (2007). "AER EAR: A matched silicon cochlea pair with address event representation interface". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 54.1, pp. 48–59.

Chicca, Elisabetta, Michael Schmuker, Martin P Nawrot, D Jaeger, and R Jung (2014). *Neuromorphic Sensors, Olfaction.*

Chou, Kenny F, Junzi Dong, H Steven Colburn, and Kamal Sen (2019). "A physiologically inspired model for solving the cocktail party problem". In: *Journal of the Association for Research in Otolaryngology* 20.6, pp. 579–593.

Christensen, Dennis Valbjørn, Regina Dittmann, Bernabé Linares-Barranco, Abu Sebastian, Manuel Le Gallo, Andrea Redaelli, Stefan Slesazeck, Thomas Mikolajick, Sabina Spiga, Stephan Menzel, et al. (2022). "2022 roadmap on neuromorphic computing and engineering". In: *Neuromorphic Computing and Engineering*.

Cramer, Benjamin, Yannik Stradmann, Johannes Schemmel, and Friedemann Zenke (2020). "The heidelberg spiking data sets for the systematic evaluation of spiking neural networks". In: *IEEE Transactions on Neural Networks and Learning Systems*.

Crespi, Alessandro, Auke Jan Ijspeert, et al. (2006). "AmphiBot II: An amphibious snake robot that crawls and swims using a central pattern generator". In: *Proceedings of the 9th international conference on climbing and walking robots (CLAWAR 2006)*. Vol. 11. 7-8, pp. 19–27.

Crespi, Alessandro, Daisy Lachat, Ariane Pasquier, and Auke Jan Ijspeert (2008a). "Controlling swimming and crawling in a fish robot using a central pattern generator". In: *Autonomous Robots* 25.1-2, pp. 3–13. ISSN: 0929-5593. DOI: 10.1007/s10514-007-9071-6. URL: http://link.springer.com/10.1007/s10514-007-9071-6.

Crespi, Alessandro, Daisy Lachat, Ariane Pasquier, and Auke Jan Ijspeert (2008b). "Controlling swimming and crawling in a fish robot using a central pattern generator". In: *Autonomous Robots* 25.1, pp. 3–13.

Cuevas-Arteaga, Brayan, Juan Pedro Dominguez-Morales, Horacio Rostro-Gonzalez, Andres Espinal, Angel F Jimenez-Fernandez, Francisco Gomez-Rodriguez, and Alejandro Linares-Barranco (2017). "A SpiNNaker application: design, implementation and validation of SCPGs". In: *International Work-Conference on Artificial Neural Networks*. Springer, pp. 548–559.

Davies, Mike et al. (2018). "Loihi: A neuromorphic manycore processor with on-chip learning". In: *IEEE Micro* 38.1, pp. 82–99.

Dávila-Chacón, Jorge, Jindong Liu, and Stefan Wermter (2018). "Enhanced robot speech recognition using biomimetic binaural sound source localization". In: *IEEE transactions on neural networks and learning systems* 30.1, pp. 138–150.

Delbruck, T. (2007). "jAER open source project". In: *Internet: http://jaer. wiki. sourceforge. net*.

Delbruck, Tobi (2008). "Frame-free dynamic digital vision". In: *Proceedings of Intl. Symp. on Secure-Life Electronics, Advanced Electronics for Quality Life and Society*, pp. 21–26.

Dominguez-Morales, Juan P, Stefano Buccelli, Daniel Gutierrez-Galan, Ilaria Colombi, Angel Jimenez-Fernandez, and Michela Chiappalone (2021a). "Real-time detection of bursts in neuronal cultures using a Neuromorphic Auditory Sensor and Spiking Neural Networks". In: *Neurocomputing* 449, pp. 422–434.

Dominguez-Morales, Juan P, D Gutierrez-Galan, A Rios-Navarro, L Duran-Lopez, M Dominguez-Morales, and A Jimenez-Fernandez (2021b). "pyNAVIS: an open-source cross-platform software for spike-based neuromorphic audio information processing". In: *Neurocomputing*.

Dominguez-Morales, Juan P, A Rios-Navarro, D Gutierrez-Galan, R Tapiador-Morales, A Jimenez-Fernandez, E Cerezuela-Escudero, M Dominguez-Morales, and A Linares-Barranco (2017a). "Live demonstration—Multilayer spiking neural network for audio samples classification using SpiNNaker". In: *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*. IEEE, pp. 1–1.

Dominguez-Morales, Juan P et al. (2017b). "Deep neural networks for the recognition and classification of heart murmurs using neuromorphic auditory sensors". In: *IEEE Trans. on Biomedical Circuits and Systems* 12.1, pp. 24–34.

Dominguez-Morales, Juan P et al. (2017c). "NAVIS: Neuromorphic Auditory VISualizer tool". In: *Neurocomputing* 237, pp. 418–422.

Dominguez-Morales, Juan P et al. (2018a). "Deep Spiking Neural Network model for time-variant signals classification: a real-time speech recognition approach". In: *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, pp. 1–8.

Domínguez Morales, Juan Pedro (2018). "Neuromorphic audio processing through real-time embedded spiking neural networks."

Dominguez-Morales, Juan Pedro, Qian Liu, Robert James, Daniel Gutierrez-Galan, Angel Jimenez-Fernandez, Simon Davidson, and Steve Furber (2018b). "Deep Spiking Neural Network model for time-variant signals classification: a real-time speech recognition approach". In: *International Joint-Conference on Neural Networks*. IEEE, pp. 45–53.

Dominguez-Morales, Juan Pedro et al. (2016). "Multilayer spiking neural network for audio samples classification using SpiNNaker". In: *International Conference on Artificial Neural Networks*. Springer, Cham, pp. 45–53.

Domínguez-Morales, M, Angel Jimenez-Fernandez, Elena Cerezuela-Escudero, Rafael Paz-Vicente, Alejandro Linares-Barranco, and Gabriel Jimenez (2011). "On the designing of spikes band-pass filters for FPGA". In: *International Conference on Artificial Neural Networks*. Springer, pp. 389–396.

Donati, Elisa, Federico Corradi, Cesare Stefanini, and Giacomo Indiveri (2014). "A spiking implementation of the lamprey's Central Pattern Generator in neuromorphic VLSI". In: *IEEE 2014 Biomedical Circuits and Systems Conference, BioCAS 2014 - Proceedings*, pp. 512–515. ISBN: 9781479923465. DOI: 10.1109/BioCAS.2014.6981775.

Douglass, John K and Nicholas J Strausfeld (1995). "Visual motion detection circuits in flies: peripheral motion computation by identified small-field retinotopic neurons". In: *Journal of Neuroscience* 15.8, pp. 5596–5611.

Drubach, Daniel (2000). *The brain explained*. Prentice Hall.

Dundur, Rekha V, MV Latte, SY Kulkarni, and MK Venkatesha (2008). "Digital filter for cochlear implant implemented on a field-programmable gate array". In: *proceedings of world academy of science, engineering and technology*. Vol. 33. Citeseer.

Duysens, Jacques and Henry W.A.A Van de Crommert (1998). "Neural control of locomotion; Part 1: The central pattern generator from cats to humans". In: *Gait & Posture* 7.2, pp. 131–141. ISSN: 0966-6362. DOI: 10.1016/S0966-6362(97)00042-8. URL: https://www.sciencedirect.com/science/article/pii/S0966636297000428.

Eshraghi, Adrien A, Ronen Nazarian, Fred F Telischi, Suhrud M Rajguru, Eric Truy, and Chhavi Gupta (2012). "The cochlear implant: historical aspects and future prospects". In: *The Anatomical Record: Advances in Integrative Anatomy and Evolutionary Biology* 295.11, pp. 1967–1980.

Falanga, Davide, Suseong Kim, and Davide Scaramuzza (2019). “How fast is too fast? the role of perception latency in high-speed sense and avoid”. In: *IEEE Robotics and Automation Letters* 4.2, pp. 1884–1891.

Falanga, Davide, Kevin Kleber, Stefano Mintchev, Dario Floreano, and Davide Scaramuzza (2018). “The foldable drone: A morphing quadrotor that can squeeze and fly”. In: *IEEE Robotics and Automation Letters* 4.2, pp. 209–216.

Fang, Ying, Zhaofei Yu, and Feng Chen (2020). “Noise Helps Optimization Escape From Saddle Points in the Synaptic Plasticity”. In: *Frontiers in neuroscience* 14, p. 343.

Finger, H., S. Liu, P. Ruvolo, and J. R. Movellan (2010). “Approaches and databases for online calibration of binaural sound localization for robotic heads”. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4340–4345. DOI: 10.1109/IROS.2010.5650515.

Finger, Holger et al. (2011). “Estimating the location of a sound source with a spike-timing localization algorithm”. In: *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*. IEEE, pp. 2461–2464.

Fischer, Tim, Marco Caversaccio, and Wilhelm Wimmer (2020). “A front-back confusion metric in horizontal sound localization: The fbc score”. In: *ACM Symposium on Applied Perception 2020*, pp. 1–5.

Fragnière, Eric (1998). *Analogue VLSI emulation of the cochlea*. Tech. rep. EPFL.

Franken, Tom P, Philip X Joris, and Philip H Smith (2018). “Principal cells of the brainstem’s interaural sound level detector are temporal differentiators rather than integrators”. In: *Elife* 7, e33854.

Frenkel, Charlotte et al. (2017). “A compact phenomenological digital neuron implementing the 20 Izhikevich behaviors”. In: *2017 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. IEEE, pp. 1–4.

Frenkel, Charlotte et al. (2019). “MorphIC: A 65-nm 738k-Synapse/mm Quad-Core Binary-Weight Digital Neuromorphic Processor with Stochastic Spike-Driven Online Learning”. In: *arXiv preprint arXiv:1904.08513*.

Fries, Pascal (2005). “A mechanism for cognitive dynamics: neuronal communication through neuronal coherence”. In: *Trends in cognitive sciences* 9.10, pp. 474–480.

Fujii, Hiroshi, Hiroyuki Ito, Kazuyuki Aihara, Natsuhiro Ichinose, and Minoru Tsukada (1996). “Dynamical cell assembly hypothesis—theoretical possibility of spatio-temporal coding in the cortex”. In: *Neural Networks* 9.8, pp. 1303–1350.

Furber, S. B. et al. (2013a). “Overview of the SpiNNaker System Architecture”. In: *IEEE Transactions on Computers* 62.12, pp. 2454–2467. ISSN: 0018-9340. DOI: 10.1109/TC.2012.142.

Furber, Stephen and Andrew Brown (2009). “Biologically-inspired massively-parallel architectures-computing beyond a million processors”. In: *Application of Concurrency to System Design, 2009. ACSD’09. Ninth International Conference On*. IEEE, pp. 3–12.

Furber, Steve (2016). "Large-scale neuromorphic computing systems". In: *Journal of neural engineering* 13.5, p. 051001.

Furber, Steve B., Francesco Galluppi, Steve Temple, and Luis A. Plana (2014b). "The SpiNNaker Project". In: *Proceedings of the IEEE*. ISSN: 00189219. DOI: 10.1109/JPROC.2014.2304638.

Furber, Steve B, Francesco Galluppi, Steve Temple, and Luis A Plana (2014a). "The SpiNNaker project". In: *Proceedings of the IEEE* 102.5, pp. 652–665.

Furber, Steve B, David R Lester, Luis A Plana, Jim D Garside, Eustace Painkras, Steve Temple, and Andrew D Brown (2013b). "Overview of the SpiNNaker system architecture". In: *IEEE Transactions on Computers* 62.12, pp. 2454–2467.

Gambin, Isabel, Ivan Grech, Owen Casha, Edward Gatt, and Joseph Micallef (2010). "Digital cochlea model implementation using Xilinx XC3S500E spartan-3E FPGA". In: *Electronics, Circuits, and Systems (ICECS), 2010 17th IEEE International Conference on*. IEEE, pp. 946–949.

Gardner, Mark B (1968). "Proximity image effect in sound localization". In: *The Journal of the Acoustical Society of America* 43.1, pp. 163–163.

George, Suma, Sihwan Kim, Sahil Shah, Jennifer Hasler, Michelle Collins, Farhan Adil, Richard Wunderlich, Stephen Nease, and Shubha Ramakrishnan (2016). "A programmable and configurable mixed-mode FPAA SoC". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 24.6, pp. 2253–2261.

Gerstner, Wulfram et al. (2002). *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press.

Glackin, Brendan, Julie A Wall, Thomas M McGinnity, Liam P Maguire, and Liam J McDaid (2010). "A spiking neural network model of the medial superior olive using spike timing dependent plasticity for sound localization". In: *Frontiers in computational neuroscience* 4, p. 18.

Glover, Arren et al. (2018). "The event-driven software library for YARP—With algorithms and iCub applications". In: *Frontiers in Robotics and AI* 4, p. 73.

Gómez-Rodríguez, F, A Jiménez-Fernández, F Pérez-Peña, L Miró, MJ Domínguez-Morales, A Ríos-Navarro, E Cerezuela, D Cascado-Caballero, and A Linares-Barranco (2016). "ED-Scorbot: A robotic test-bed framework for FPGA-based neuromorphic systems". In: *Biomedical Robotics and Biomechatronics (BioRob), 2016 6th IEEE International Conference on*. IEEE, pp. 237–242.

Gomez-Rodriguez, F. et al. (2005). "Two Hardware Implementations of the Exhaustive Synthetic AER Generation Method". In: *Computational Intelligence and Bioinspired Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 534–540. ISBN: 978-3-540-32106-4.

Gomez-Rodriguez, Francisco, Alejandro Linares-Barranco, L Miro, Shih-Chii Liu, André Van Schaik, Ralph Etienne-Cummings, and M Anthony Lewis (2007). "AER auditory filtering and CPG for robot control". In: *2007 IEEE International Symposium on Circuits and Systems*. IEEE, pp. 1201–1204.

Gonzalez-Billandon, Jonas, Alessandra Sciutti, Matthew Tata, Giulio Sandini, and Francesco Rea (2020). "Audiovisual cognitive architecture for autonomous learning of face localisation by a Humanoid Robot". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 5979–5985.

Grasse, Lukas, Sylvain J Boutros, and Matthew S Tata (2021). "Speech interaction to control a hands-free delivery robot for high-risk health care scenarios". In: *Frontiers in Robotics and AI* 8.

Grillner, Sten, Peter Wallén, Kazuya Saitoh, Alexander Kozlov, and Brita Robertson (2008). "Neural bases of goal-directed locomotion in vertebrates—an overview". In: *Brain research reviews* 57.1, pp. 2–12.

Grothe, Benedikt, Michael Pecka, and David McAlpine (2010). "Mechanisms of sound localization in mammals". In: *Physiological reviews* 90.3, pp. 983–1012.

Guertin, Pierre A. (2009). "The mammalian central pattern generator for locomotion". In: *Brain Research Reviews* 62.1, pp. 45–56. ISSN: 0165-0173. DOI: 10.1016/J.BRAINRESREV.2009.08.002. URL: https://www.sciencedirect.com/science/article/pii/S0165017309000812.

Gutierrez-Galan, Daniel, Chiara Bartolozzi, Juan Pedro Dominguez-Morales, Angel Jimenez-Fernandez, and Alejandro Linares-Barranco (2022). "Towards the Neuromorphic Implementation of the Auditory Perception in the ICub Robotic Platform". In: *Neuro-Inspired Computational Elements Conference*. NICE 2022. Association for Computing Machinery, 11–12. ISBN: 9781450395595. DOI: 10.1145/3517343.3517347. URL: https://doi.org/10.1145/3517343.3517347.

Gutierrez-Galan, Daniel, Juan P Dominguez-Morales, Fernando Perez-Pena, Angel Jimenez-Fernandez, and Alejandro Linares-Barranco (2019a). "Live demonstration: neuromorphic robotics, from audio to locomotion through spiking CPG on SpiNNaker". In: *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, pp. 1–1.

Gutierrez-Galan, Daniel, Juan P Dominguez-Morales, Fernando Perez-Peña, Angel Jimenez-Fernandez, and Alejandro Linares-Barranco (2020). "NeuroPod: a real-time neuromorphic spiking CPG applied to robotics". In: *Neurocomputing* 381, pp. 10–19.

Gutierrez-Galan, Daniel, Juan Pedro Dominguez-Morales, Angel Jimenez-Fernandez, Ricardo Tapiador-Morales, Antonio Rios-Navarro, and Alejandro Linares-Barranco (2019b). "A neuromorphic approach of the sound source localization task in real-time embedded systems: work-in-progress". In: *Proceedings of the International Conference on Embedded Software Companion*, pp. 1–2.

Gutierrez-Galan, Daniel, Thorben Schoepe, Juan P Dominguez-Morales, Angel Jimenez-Fernandez, Elisabetta Chicca, and Alejandro Linares-Barranco (2021a). "An event-based digital time difference encoder model implementation for neuromorphic systems". In: *IEEE Transactions on Neural Networks and Learning Systems*.

Gutierrez-Galan, Daniel et al. (2021b). "OpenNAS: Open Source Neuromorphic Auditory Sensor HDL code generator for FPGA implementations". In: *Neurocomputing* 436, pp. 35–38.

Habets, Emanuel AP (2006). "Room impulse response generator". In: *Technische Universiteit Eindhoven, Tech. Rep* 2.2.4, p. 1.

Haessig, Germain et al. (2020). "Event-based computation for touch localization based on precise spike timing". In: *Frontiers in Neuroscience* 14, p. 420.

Hambrook, Dillon A et al. (2017). "A Bayesian computational basis for auditory selective attention using head rotation and the interaural time-difference cue". In: *PloS one* 12.10.

Hamilton, Tara Julia, Craig Jin, Andre Van Schaik, and Jonathan Tapson (2008). "An active 2-D silicon cochlea". In: *IEEE Transactions on biomedical circuits and systems* 2.1, pp. 30–43.

Hassentstein, B and W Reichardt (Jan. 1956). "Systemtheoretische Analyse der Zeit-, Reihenfolgen- und Vorzeichenauswertung bei der Bewegungsperzeption des Rüsselkäfers Chlorophanus". In: *Z. Naturforsch.* 11b, pp. 513–524. DOI: 10.1515/znb-1956-9-1004.

Hodgkin, Alan L and Andrew F Huxley (1939). "Action potentials recorded from inside a nerve fibre". In: *Nature* 144.3651, p. 710.

Horiuchi, T. K. (2009). "A Spike-Latency Model for Sonar-Based Navigation in Obstacle Fields". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 56.11, pp. 2393–2401. ISSN: 1549-8328. DOI: 10.1109/TCSI.2009.2015597.

Hull, Kerry L (2011). *Human Form, Human Function: Essentials of Anatomy & Physiology*. Lippincott Williams & Wilkins.

Hynna, Kai and Kwabena Boahen (2001). "Space-rate coding in an adaptive silicon neuron". In: *Neural Networks* 14.6-7, pp. 645–656.

Iacono, Massimiliano, Stefan Weber, Arren Glover, and Chiara Bartolozzi (2018). "Towards event-driven object detection with off-the-shelf deep learning". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 1–9.

Iakymchuk, Taras, Alfredo Rosado, Teresa Serrano-Gotarredona, Bernabé Linares-Barranco, Angel Jiménez-Fernandez, Alejandro Linares-Barranco, and Gabriel Jiménez-Moreno (2014). "An AER handshake-less modular infrastructure PCB with x8 2.5 Gbps LVDS serial links". In: *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, pp. 1556–1559.

Indiveri, G. et al. (2011a). "Neuromorphic silicon neuron circuits". In: *Frontiers in Neuroscience* 5, pp. 1–23. ISSN: 1662-453X. DOI: 10.3389/fnins.2011.00073.

Indiveri, Giacomo, Elisabetta Chicca, and Rodney Douglas (2006). "A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity". In: *IEEE transactions on neural networks* 17.1, pp. 211–221.

Indiveri, Giacomo et al. (2011b). "Neuromorphic silicon neuron circuits". In: *Frontiers in neuroscience* 5, p. 73.

Ito, Akira (1989). *Barrel shifter*. US Patent 4,829,460.

Jäckel, David, Rico Moeckel, and Shih-Chii Liu (2010). "Sound recognition with spiking silicon cochlea and Hidden Markov Models". In: *Ph. D. Research in Microelectronics and Electronics (PRIME), 2010 Conference on*. IEEE, pp. 1–4.

James, Robert (2020). "Spikes from sound: A model of the human auditory periphery on SpiNNaker". In:

Janus, Scott (2004). *Audio in the 21st Century*. Intel Press.

Jeffress, Lloyd A (1948). "A place theory of sound localization." In: *Journal of comparative and physiological psychology* 41.1, p. 35.

Jimenez-Fernandez, Angel, Gabriel Jimenez-Moreno, Alejandro Linares-Barranco, Manuel J Dominguez-Morales, Rafael Paz-Vicente, and Anton Civit-Balcells (2012). "A neuro-inspired spike-based PID motor controller for multi-motor robots with low cost FPGAs". In: *Sensors* 12.4, pp. 3831–3856.

Jimenez-Fernandez, Angel et al. (2010). "Building blocks for spikes signals processing". In: *Neural Networks (IJCNN), The 2010 International Joint Conference on*. IEEE, pp. 1–8.

Jimenez-Fernandez, Angel et al. (2017). "A Binaural Neuromorphic Auditory Sensor for FPGA: A Spike Signal Processing Approach." In: *IEEE Trans. Neural Netw. Learning Syst.* 28.4, pp. 804–818.

Jiménez Fernández, Ángel Francisco (2010). "Diseño y evaluación de sistemas de control y procesamiento de señales basados en modelos neuronales pulsantes". In:

Johnston, Daniel and Samuel Miao-Sin Wu (1994). *Foundations of cellular neurophysiology*. MIT press.

Jones, Simon, Ray Meddis, Seow Chuan Lim, and A Robert Temple (2000). "Toward a digital neuromorphic pitch extraction system". In: *IEEE transactions on neural networks* 11.4, pp. 978–987.

Kaas, John H. (1997). "Topographic Maps are Fundamental to Sensory Processing". In: *Brain Research Bulletin* 44.2, pp. 107–112.

Kandel, Eric R, James H Schwartz, Thomas M Jessell, Steven Siegelbaum, A James Hudspeth, and Sarah Mack (2000). *Principles of neural science*. Vol. 4. McGraw-hill New York.

Kandler, Karl, Amanda Clause, and Jihyun Noh (2009). "Tonotopic reorganization of developing auditory brainstem circuits". In: *Nature neuroscience* 12.6, pp. 711–717.

Katsiamis, Andreas G, Emmanuel M Drakakis, and Richard F Lyon (2007). "Practical gammatone-like filters for auditory processing". In: *EURASIP Journal on Audio, Speech, and Music Processing* 2007.1, p. 063685.

Khacef, Lyes, Laurent Rodriguez, and Benoît Miramond (2020). "Brain-inspired self-organization with cellular neuromorphic computing for multimodal unsupervised learning". In: *Electronics* 9.10, p. 1605.

Kish, Lazlo B. (2002). "End of Moore's law: thermal (noise) death of integration in microand nano electronics". In: *Physics Letters A* 305, pp. 144–149.

Knight, James and Thomas Nowotny (2019). “GeNN: GPU-enhanced neural networks”. In: *NEST Conference 2019*.

Knight, James C and Thomas Nowotny (2018). “GPUs outperform current HPC and neuromorphic solutions in terms of speed and energy when simulating a highly-connected cortical model”. In: *Frontiers in neuroscience* 12, p. 941.

Knight, James C and Thomas Nowotny (2021). “Larger GPU-accelerated brain simulations with procedural connectivity”. In: *Nature Computational Science* 1.2, pp. 136–142.

Kumar, Nagendra, Wolfgang Himmelbauer, Gert Cauwenberghs, and Andreas G Andreou (1998). “An analog VLSI chip with asynchronous interface for auditory feature extraction”. In: *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 45.5, pp. 600–606.

Lande, Tor Sverre, J-T Marienborg, and Yngvar Berg (2000). “Neuromorphic cochlea implants”. In: *2000 IEEE International Symposium on Circuits and Systems (ISCAS)*. Vol. 4. IEEE, pp. 401–404.

Lazzaro, John and Carver Mead (1989a). “Circuit models of sensory transduction in the cochlea”. In: *Analog VLSI Implementation of Neural Systems*. Springer, pp. 85–101.

Lazzaro, John and Carver A Mead (1989b). “A silicon model of auditory localization”. In: *Neural computation* 1.1, pp. 47–57.

Lazzaro, John and John Wawrzynek (1995). “A multi-sender asynchronous extension to the AER protocol”. In: *Advanced Research in VLSI, 1995. Proceedings., Sixteenth Conference on*. IEEE, pp. 158–169.

Lazzaro, John, John Wawrzynek, Misha Mahowald, Massimo Sivilotti, and Dave Gillespie (1993). “Silicon auditory processors as computer peripherals”. In: *Advances in Neural Information Processing Systems*, pp. 820–827.

Lele, Ashwin, Yan Fang, Justin Ting, and Arijit Raychowdhury (2021). “An End-to-end Spiking Neural Network Platform for Edge Robotics: From Event-Cameras to Central Pattern Generation”. In: *IEEE Transactions on Cognitive and Developmental Systems*.

Leong, Monk-Ping, Craig T Jin, and Philip HW Leong (2003). “An FPGA-based electronic cochlea”. In: *EURASIP Journal on Applied Signal Processing* 2003, pp. 629–638.

Levi, Timothée et al. (2018). “Digital implementation of Hodgkin–Huxley neuron model for neurological diseases studies”. In: *Artificial Life and Robotics*.

Lichtsteiner, Patrick et al. (2008). “A 128$\times$128 120 dB 15μs Latency Asynchronous Temporal Contrast Vision Sensor”. In: *IEEE journal of solid-state circuits* 43.2, pp. 566–576.

Linares-Barranco, A. et al. (2006). “On algorithmic rate-coded AER generation”. In: *IEEE Transactions on Neural Networks* 17.3, pp. 771–788.

Linares-Barranco, Alejandro, Fernando Perez-Peña, Angel Jimenez-Fernandez, and Elisabetta Chicca (2020). “ED-BioRob: a Neuromorphic Robotic Arm with FPGA-based infrastructure for Bio-inspired spiking motor controllers”. In: *Frontiers in Neurorobotics* 14.

Liu, Jindong, Harry Erwin, and Stefan Wermter (2008). "Mobile robot broadband sound localisation using a biologically inspired spiking neural network". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 2191–2196.

Liu, Jindong, David Perez-Gonzalez, Adrian Rees, Harry Erwin, and Stefan Wermter (2010a). "A biologically inspired spiking neural network model of the auditory midbrain for sound source localisation". In: *Neurocomputing* 74.1-3, pp. 129–139.

Liu, Qian, Cameron Patterson, Steve Furber, Zhangqin Huang, Yibin Hou, and Huibing Zhang (2013). "Modeling populations of spiking neurons for fine timing sound localization". In: *The 2013 International Joint Conference on Neural Networks (IJCNN)*. IEEE, pp. 1–8.

Liu, Shih-Chii, Tobi Delbruck, Giacomo Indiveri, Adrian Whatley, and Rodney Douglas (2015). *Event-based neuromorphic systems*. John Wiley & Sons.

Liu, Shih-Chii, Jörg Kramer, Giacomo Indiveri, Tobias Delbrück, and Rodney Douglas (2002). *Analog VLSI: circuits and principles*. MIT press.

Liu, Shih-Chii, André van Schaik, Bradley A Minch, and Tobi Delbruck (2014). "Asynchronous Binaural Spatial Audition Sensor With 2x64x4 Channel Output". In: *IEEE transactions on biomedical circuits and systems* 8.4, pp. 453–464.

Liu, Shih-Chii, André Van Schaik, Bradley A Mincti, and Tobi Delbruck (2010b). "Event-based 64-channel binaural silicon cochlea with Q enhancement mechanisms". In: *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*. IEEE, pp. 2027–2030.

Liu, Weimin, Andreas G Andreou, and MH Goldstein (1991). "An analog integrated speech front-end based on the auditory periphery". In: *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*. Vol. 2. IEEE, pp. 861–864.

Liu, Weimin, Andreas G Andreou, and Moise H Goldstein (1992). "Voiced-speech representation by an analog silicon model of the auditory periphery". In: *IEEE Transactions on Neural Networks* 3.3, pp. 477–487.

Liu, Ying, Khaled Benkrid, AbdSamad Benkrid, and Server Kasap (2009). "An fpga-based web server for high performance biological sequence alignment". In: *2009 NASA/ESA Conference on Adaptive Hardware and Systems*. IEEE, pp. 361–368.

Lopez-Poveda, Enrique A (2018). "Olivocochlear efferents in animals and humans: from anatomy to clinical relevance". In: *Frontiers in neurology* 9, p. 197.

Lyon, Richard (1982). "A computational model of filtering, detection, and compression in the cochlea". In: *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'82.* Vol. 7. IEEE, pp. 1282–1285.

Lyon, Richard F (2017). *Human and machine hearing*. Cambridge University Press.

Lyon, Richard F and Carver Mead (1988). "An analog electronic cochlea". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 36.7, pp. 1119–1134.

Maass, Wolfgang and Christopher M Bishop (2001). *Pulsed neural networks*. MIT press.

Mahowald, Misha (1992). "VLSI analogs of neuronal visual processing: a synthesis of form and function".

Maqueda, Ana I, Antonio Loquercio, Guillermo Gallego, Narciso García, and Davide Scaramuzza (2018). "Event-based vision meets deep learning on steering prediction for self-driving cars". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5419–5427.

Massaro, Dominic W and David G Stork (1998). "Speech recognition and sensory integration: a 240-year-old theorem helps explain how people and machines can integrate auditory and visual information to understand speech". In: *American Scientist* 86.3, pp. 236–244.

Massoud, Tarek M and Timothy K Horiuchi (2010). "A neuromorphic VLSI head direction cell system". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 58.1, pp. 150–163.

McDonald, John J, Wolfgang A Teder-SaÈlejaÈrvi, and Steven A Hillyard (2000). "Involuntary orienting to sound improves visual perception". In: *Nature* 407.6806, pp. 906–908.

Mead, Carver (1989). *Analog VLSI and Neural Systems*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0-201-05992-4.

Mead, Carver (1990). "Neuromorphic electronic systems". In: *Proceedings of the IEEE* 78.10, pp. 1629–1636.

Metta, Giorgio, Paul Fitzpatrick, and Lorenzo Natale (2006). "YARP: yet another robot platform". In: *International Journal of Advanced Robotic Systems* 3.1, p. 8.

Milde, Moritz B, Olivier JN Bertrand, Ryad Benosman, Martin Egelhaaf, and Elisabetta Chicca (2015). "Bioinspired event-driven collision avoidance algorithm based on optic flow". In: *Event-based Control, Communication, and Signal Processing (EBCCSP), 2015 International Conference on*. IEEE, pp. 1–7.

Milde, Moritz B, Hermann Blum, Alexander Dietmüller, Dora Sumislawska, Jörg Conradt, Giacomo Indiveri, and Yulia Sandamirskaya (2017). "Obstacle avoidance and target acquisition for robot navigation using a mixed signal analog/digital neuromorphic processing system". In: *Frontiers in neurorobotics* 11, p. 28.

Milde, Moritz B et al. (2018). "Spiking elementary motion detector in neuromorphic systems". In: *Neural computation* 30.9, pp. 2384–2417.

Minassian, Karen, Ursula S Hofstoetter, Florin Dzeladini, Pierre A Guertin, and Auke Ijspeert (2017). "The Human Central Pattern Generator for Locomotion: Does It Exist and Contribute to Walking?" In: *The Neuroscientist* 23.6, pp. 649–663.

Miró Amarante, María Lourdes (2013). "Una aportación al procesado neuromórfico de audio basado en modelos pulsantes. Desde la cóclea a la percepción auditiva".

Molholm, Sophie and John J Foxe (2005). "Look 'hear', primary auditory cortex is active during lip-reading". In: *Neuroreport* 16.2, pp. 123–124.

Monforte, Marco, Ander Arriandiaga, Arren Glover, and Chiara Bartolozzi (2020). "Where and When: Event-Based Spatiotemporal Trajectory Prediction from the iCub's Point-Of-View". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 9521–9527.

Montealegre, Norma, David Merodio, Agustin Fernandez, and Philippe Armbruster (2015). "In-flight reconfigurable FPGA-based space systems". In: *2015 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. IEEE, pp. 1–8.

Moradi, Saber, Ning Qiao, Fabio Stefanini, and Giacomo Indiveri (2017). "A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs)". In: *IEEE transactions on biomedical circuits and systems* 12.1, pp. 106–122.

Moravec, Hans (1998). "When will computer hardware match the human brain?" In: *Jouranl of Evolution and Technology* 1.

Mugliette, Christian, Ivan Grech, Owen Casha, Edward Gatt, and Joseph Micallef (2011). "FPGA active digital cochlea model". In: *Electronics, Circuits and Systems (ICECS), 2011 18th IEEE International Conference on*. IEEE, pp. 699–702.

Müller, G. R. and J. Conradt (2011). "A miniature low-power sensor system for real time 2D visual tracking of LED markers". In: *2011 IEEE International Conference on Robotics and Biomimetics*, pp. 2429–2434. DOI: 10.1109/ROBIO.2011.6181669.

Natale, Lorenzo, Chiara Bartolozzi, Francesco Nori, Giulio Sandini, and Giorgio Metta (2021). "The iCub platform: evolution and current trends". In: *arXiv e-prints*, arXiv–2105.

Natale, Lorenzo, Chiara Bartolozzi, Daniele Pucci, Agnieszka Wykowska, and Giorgio Metta (2017). "icub: The not-yet-finished story of building a robot child". In: *Science Robotics* 2.13.

Nathan, Adam (2006). *Windows presentation foundation unleashed*. Pearson Education.

Naveros, Francisco, Niceto R Luque, Eduardo Ros, and Angelo Arleo (2019). "VOR adaptation on a humanoid iCub robot using a spiking cerebellar model". In: *IEEE transactions on cybernetics* 50.11, pp. 4744–4757.

Oster, Matthias, Rodney Douglas, and Shih-Chii Liu (July 2009). "Computation with Spikes in a Winner-Take-All Network". In: *Neural computation* 21, pp. 2437–65. DOI: 10.1162/neco.2009.07-08-829.

Painkras, Eustace et al. (2013). "SpiNNaker: A 1-W 18-core system-on-chip for massively-parallel neural network simulation". In: *IEEE Journal of Solid-State Circuits* 48.8, pp. 1943–1953.

Park, Paul KJ et al. (2013). "Fast neuromorphic sound localization for binaural hearing aids". In: *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE.

Parmiggiani, Alberto, Marco Maggiali, Lorenzo Natale, Francesco Nori, Alexander Schmitz, Nikos Tsagarakis, Jose Santos Victor, Francesco Becchi, Giulio Sandini, and Giorgio Metta (2012). “The design of the iCub humanoid robot”. In: *International journal of humanoid robotics* 9.04, p. 1250027.

Pasquale, Giulia, Carlo Ciliberto, Francesca Odone, Lorenzo Rosasco, and Lorenzo Natale (2015). “Teaching iCub to recognize objects using deep Convolutional Neural Networks”. In: *Machine Learning for Interactive Systems*. PMLR, pp. 21–25.

Patton, Kevin T, Gary A Thibodeau, and Matthew M Douglas (2012). *Essentials of anatomy & physiology*. Elsevier/Mosby.

Pavlou, Athanasios and Matthew Casey (2010). “Simulating the effects of cortical feedback in the superior colliculus with topographic maps”. In: *The 2010 International Joint Conference on Neural Networks (IJCNN)*. IEEE, pp. 1–8.

Perez-Peña, Fernando, Arturo Morgado-Estevez, Alejandro Linares-Barranco, Angel Jimenez-Fernandez, Francisco Gomez-Rodriguez, Gabriel Jimenez-Moreno, and Juan Lopez-Coronado (2013a). “Neuro-inspired spike-based motion: from dynamic vision sensor to robot motor open-loop control through spike-VITE”. In: *Sensors* 13.11, pp. 15805–15832.

Perez-Peña, Fernando, Arturo Morgado-Estevez, Alejandro Linares-Barranco, Angel Jimenez-Fernandez, Francisco Gomez-Rodriguez, Gabriel Jimenez-Moreno, and Juan Lopez-Coronado (2013b). “Neuro-Inspired Spike-Based Motion: From Dynamic Vision Sensor to Robot Motor Open-Loop Control through Spike-VITE”. In: *Sensors* 13.11, 15805–15832. ISSN: 1424-8220.

Perez-Peña, Fernando et al. (2019). “Digital neuromorphic real-time platform”. In: *Neurocomputing*.

Plana, L.A., J. Heathcote, J.S. Pepper, S. Davidson, J. Garside, S. Temple, and S.B. Furber (2014). “spI/O: A library of FPGA designs and reusable modules for I/O in SpiNNaker systems”. In: DOI: 10.5281/zenodo.51476.

Plana, Luis A, Steve B Furber, Steve Temple, Mukaram Khan, Yebin Shi, Jian Wu, and Shufan Yang (2007). “A GALS infrastructure for a massively parallel multiprocessor”. In: *IEEE Design & Test of Computers* 24.5.

Plenge, Georg (1974). “On the differences between localization and lateralization”. In: *The Journal of the Acoustical Society of America* 56.3, pp. 944–951.

Posch, Christoph, Daniel Matolin, and Rainer Wohlgenannt (2010). “A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS”. In: *IEEE Journal of Solid-State Circuits* 46.1, pp. 259–275.

Qiao, Ning, Hesham Mostafa, Federico Corradi, Marc Osswald, Fabio Stefanini, Dora Sumislawska, and Giacomo Indiveri (2015). “A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses”. In: *Frontiers in neuroscience* 9, p. 141.

Rakic, Pasko (1988). “Specification of cerebral cortical areas”. In: *Science* 241.4862, pp. 170–176.

Rasetto, Marco, Juan P Dominguez-Morales, Angel Jimenez-Fernandez, and Ryad Benosman (2021). “Event Based Time-Vectors for auditory features extraction: a neuromorphic approach for low power audio recognition". In: *arXiv preprint arXiv:2112.07011*.

Rigden, J (1996). *Body, physics of*.

Rios-Navarro, Antonio, Juan Pedro Dominguez-Morales, Ricardo Tapiador-Morales, Daniel Gutierrez-Galan, Angel Jimenez-Fernandez, and Alejandro Linares-Barranco (2016). "A 20Mevps/32Mev event-based USB framework for neuromorphic systems debugging". In: *2016 Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*. IEEE, pp. 1–6.

Risoud, M., J.-N. Hanson, F. Gauvrit, C. Renard, P.-E. Lemesre, N.-X. Bonne, and C. Vincent (2018). "Sound source localization". In: *European Annals of Otorhinolaryngology, Head and Neck Diseases* 135.4, pp. 259–264. ISSN: 1879-7296. DOI: https://doi.org/10.1016/j.anorl.2018.04.009. URL: https://www.sciencedirect.com/science/article/pii/S187972961830067X.

Rodríguez Valiente, A, A Trinidad, JR García Berrocal, C Górriz, and R Ramírez Camacho (2014). "Extended high-frequency (9–20 kHz) audiometry reference thresholds in 645 healthy subjects". In: *International journal of audiology* 53.8, pp. 531–545.

Rostro-Gonzalez, Horacio, Pedro Alberto Cerna-Garcia, Gerardo Trejo-Caballero, Carlos H Garcia-Capulin, Mario Alberto Ibarra-Manzano, Juan Gabriel Avina-Cervantes, and César Torres-Huitzil (2015). "A CPG system based on spiking neurons for hexapod robot locomotion". In: *Neurocomputing* 170, pp. 47–54.

Sabena, Davide, Luca Sterpone, Mario Schölzel, Tobias Koal, Heinrich Theodor Vierhaus, S Wong, Robért Glein, Florian Rittner, C Stender, Mario Porrmann, et al. (2014). "Reconfigurable high performance architectures: How much are they ready for safety-critical applications?" In: *2014 19th IEEE European Test Symposium (ETS)*. IEEE, pp. 1–8.

Sartoretti, Guillaume, Samuel Shaw, Katie Lam, Naixin Fan, Matthew Travers, and Howie Choset (2018). "Central Pattern Generator with Inertial Feedback for Stable Locomotion and Climbing in Unstructured Terrain". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1–5.

Schaik, André van (2010). "Adaptive sound localization with a silicon cochlea pair". In: *Frontiers in neuroscience* 4, p. 196.

Schemmel, Johannes, Daniel Brüderle, Andreas Grübl, Matthias Hock, Karlheinz Meier, and Sebastian Millner (2010). "A wafer-scale neuromorphic hardware system for large-scale neural modeling". In: *2010 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, pp. 1947–1950.

Schilling, Malte, Thierry Hoinville, Josef Schmitz, and Holk Cruse (2013). "Walknet, a bio-inspired controller for hexapod walking". In: *Biological Cybernetics* 107.4, pp. 397–419. ISSN: 03401200. DOI: 10.1007/s00422-013-0563-5.

Schneider, Axel, Jan Paskarbeit, Mattias Schaeffersmann, and Josef Schmitz (2012). "Hector, a new hexapod robot platform with increased mobilitycontrol approach, design and communication". In: *Advances in Autonomous Mini Robots*. Springer, pp. 249–264.

Schnupp, Jan, Israel Nelken, and Andrew King (2011). *Auditory neuroscience: Making sense of sound*. MIT press.

Schoepe, Thorben, Daniel Gutierrez-Galan, Juan Pedro Dominguez-Morales, Angel Jimenez-Fernandez, Alejandro Linares-Barranco, and Elisabetta Chicca (2019). "Neuromorphic sensory integration for combining sound source localization and collision avoidance". In: *2019 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. IEEE, pp. 1–4.

Schoepe, Thorben, Daniel Gutierrez-Galan, Juan Pedro Dominguez-Morales, Angel Jimenez-Fernandez, Alejandro Linares-Barranco, and Elisabetta Chicca (2020). "Live Demonstration: Neuromorphic Sensory Integration for Combining Sound Source Localization and Collision Avoidance". In: *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, pp. 1–1.

Schreiner, Christoph E and Jeffery A Winer (2005). *The inferior colliculus*. Springer.

Serrano-Gotarredona, Rafael, Matthias Oster, Patrick Lichtsteiner, Alejandro Linares-Barranco, Rafael Paz-Vicente, Francisco Gómez-Rodríguez, Luis Camuñas-Mesa, Raphael Berner, Manuel Rivas-Pérez, Tobi Delbruck, et al. (2009). "CAVIAR: A 45k neuron, 5M synapse, 12G connects/s AER hardware sensory–processing–learning–actuating system for high-speed visual object recognition and tracking". In: *IEEE Transactions on Neural Networks* 20.9, pp. 1417–1438.

Serrano-Gotarredona, Teresa and Bernabé Linares-Barranco (2013). "A 128 $\times$ 128 1.5% Contrast Sensitivity 0.9% FPN 3 μs Latency 4 mW Asynchronous Frame-Free Dynamic Vision Sensor Using Transimpedance Preamplifiers". In: *IEEE Journal of Solid-State Circuits* 48.3, pp. 827–838.

Shadlen, Michael N and William T Newsome (1994). "Noise, neural codes and cortical organization". In: *Current opinion in neurobiology* 4.4, pp. 569–579.

Shaikh, Danish, John Hallam, and Jakob Christensen-Dalsgaard (2016). "From "ear" to there: a review of biorobotic models of auditory processing in lizards". In: *Biological cybernetics* 110.4, pp. 303–317.

Shepherd, Gordon M (2003). *The synaptic organization of the brain*. Oxford University Press.

Shi, Zhuanghua and Hermann Müller (Nov. 2013). "Multisensory perception and action: Development, decision-making, and neural mechanisms". In: *Frontiers in integrative neuroscience* 7, p. 81. DOI: 10.3389/fnint.2013.00081.

Shiraishi, Hisako (2003). "Design of an analog VLSI cochlea".

Shore, SE (2009). *Auditory/somatosensory interactions*. Elsevier.

Sivilotti, Massimo Antonio (1991). *Wiring considerations in analog VLSI systems, with application to field-programmable networks.*

Skottun, Bernt C, Trevor M Shackleton, Robert H Arnott, and Alan R Palmer (2001). "The ability of inferior colliculus neurons to signal differences in interaural delay". In: *Proceedings of the National Academy of Sciences* 98.24, pp. 14050–14054.

Soleimani, Hamid et al. (2012). “Biologically inspired spiking neurons: Piecewise linear models and digital implementation”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers*.

Still, Susanne, Bernhard Schölkopf, Klaus Hepp, and Rodney J Douglas (2001). “Four-legged walking gait control using a neuromorphic chip interfaced to a support vector learning algorithm”. In: *Advances in neural information processing systems*, pp. 741–747.

Sugiarto, I., P. Campos, N. Dahir, G. Tempesti, and S. Furber (2017). “Optimized task graph mapping on a many-core neuromorphic supercomputer”. In: *2017 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7. DOI: 10.1109/HPEC.2017.8091066.

Summerfield, Clive D and Richard F Lyon (1992). “ASIC implementation of the Lyon cochlea model”. In: *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*. Vol. 5. IEEE, pp. 673–676.

Tabibi, Sonia, Andrea Kegel, Wai Kong Lai, and Norbert Dillier (2017). “Investigating the use of a Gammatone filterbank for a cochlear implant coding strategy”. In: *Journal of Neuroscience Methods* 277, pp. 63–74. ISSN: 0165-0270. DOI: https://doi.org/10.1016/j.jneumeth.2016.12.004. URL: https://www.sciencedirect.com/science/article/pii/S0165027016302898.

Tapiador-Morales, Ricardo et al. (2018). “Neuromorphic LIF Row-by-Row Multiconvolution Processor for FPGA”. In: *IEEE transactions on biomedical circuits and systems* 13.1, pp. 159–169.

Thakur, Chetan Singh, Tara Julia Hamilton, Jonathan Tapson, André van Schaik, and Richard F Lyon (2014). “FPGA Implementation of the CAR Model of the Cochlea”. In: *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*. IEEE, pp. 1853–1856.

Thakur, Chetan Singh, Runchun Mark Wang, Saeed Afshar, Tara Julia Hamilton, Jonathan Tapson, Shihab Shamma, and André van Schaik (2015). “Sound stream segregation: a neuromorphic approach to solve the “cocktail party problem” in real-time”. In: *Frontiers in neuroscience* 9, p. 309.

Thor, Mathias, Beck Strohmer, and Poramate Manoonpong (2021). “Locomotion control with frequency and motor pattern adaptations”. In: *Frontiers in Neural Circuits* 15.

Thorpe, Simon J, Adrien Brilhault, and José-Antonio Perez-Carrasco (2010). “Suggestions for a biologically inspired spiking retina using order-based coding”. In: *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*. IEEE, pp. 265–268.

Tikhanoff, Vadim et al. (2010). “Integration of speech and action in humanoid robots: iCub simulation experiments”. In: *IEEE Transactions on Autonomous Mental Development* 3.1, pp. 17–29.

Ting, Justin, Yan Fang, Ashwin Lele, and Arijit Raychowdhury (2020). “Bio-inspired gait imitation of hexapod robot using event-based vision sensor and spiking neural network”. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, pp. 1–7.

Van Schaik, André and Eric Fragnière (2001). “Pseudo-voltage domain implementation of a 2-dimensional silicon cochlea”. In: *Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on*. Vol. 3. IEEE, pp. 185–188.

Van Schaik, André, Eric Fragnière, and Eric A Vittoz (1996). “Improved silicon cochlea using compatible lateral bipolar transistors”. In: *Advances in neural information processing systems*, pp. 671–677.

Van Schaik, André et al. (2004). “A neuromorphic sound localizer for a smart MEMS system”. In: *Analog Integrated Circuits and Signal Processing*.

Vanarse, Anup (2020). “Neuronose: An empirical study of neuromorphic approaches for the development of an artificial olfactory system”. In:

Vogelstein, R Jacob, Francesco VG Tenore, Lisa Guevremont, Ralph Etienne-Cummings, and Vivian K Mushahwar (2008). “A silicon central pattern generator controls locomotion in vivo”. In: *IEEE transactions on biomedical circuits and systems* 2.3, pp. 212–222.

Voutsas, Kyriakos and Jürgen Adamy (2007). “A biologically inspired spiking neural network for sound source lateralization”. In: *IEEE transactions on Neural Networks* 18.6, pp. 1785–1799.

Wallach, Hans (1939). “On sound localization”. In: *The Journal of the Acoustical Society of America* 10.4, pp. 270–274.

Ward-Cherrier, Benjamin, Nicholas Pestell, and Nathan F Lepora (2020). “NeuroTac: A neuromorphic optical tactile sensor applied to texture recognition”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 2654–2660.

Warden, Pete (2018). “Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition”. In: *CoRR* abs/1804.03209. arXiv: 1804.03209. URL: http://arxiv.org/abs/1804.03209.

Warren, Richard M (2013). *Auditory perception: A new synthesis*. Vol. 109. Elsevier.

Watts, Lloyd, Douglas A Kerns, Richard F Lyon, and Carver A Mead (1992). “Improved implementation of the silicon cochlea”. In: *IEEE Journal of Solid-state circuits* 27.5, pp. 692–700.

Wen, Bo and Kwabena Boahen (2009). “A silicon cochlea with active coupling”. In: *IEEE transactions on biomedical circuits and systems* 3.6, pp. 444–455.

Westerman, Wayne C, David PM Northmore, and John G Elias (1997). “Neuromorphic synapses for artificial dendrites”. In: *Analog Integrated Circuits and Signal Processing* 13.1-2, pp. 167–184.

Wiesmann, Georg, Stephan Schraml, Martin Litzenberger, Ahmed Nabil Belbachir, Michael Hofstätter, and Chiara Bartolozzi (2012). “Event-driven embodied system for feature extraction and object recognition in robotic applications”. In: *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, pp. 76–82.

Wilson, Blake S, Charles C Finley, Dewey T Lawson, Robert D Wolford, Donald K Eddington, and William M Rabinowitz (1991). "Better speech recognition with cochlear implants". In: *Nature* 352.6332, pp. 236–238.

Xu, Ying, Chetan S Thakur, Ram K Singh, Tara Julia Hamilton, Runchun M Wang, and André van Schaik (2018a). "A FPGA implementation of the CAR-FAC cochlear model". In: *Frontiers in neuroscience* 12, p. 198.

Xu, Ying et al. (2018b). "A FPGA implementation of the CAR-FAC cochlear model". In: *Frontiers in neuroscience* 12, p. 198.

Yang, M. et al. (2016). "A 0.5V 55 μW 64x2-channel binaural silicon cochlea for event-driven stereo-audio sensing". In: *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 388–389. DOI: 10.1109/ISSCC.2016.7418070.

Yang, Minhao, Chen-Han Chien, Tobi Delbruck, and Shih-Chii Liu (2016). "A 0.5 V 55μW 64$\times$2 Channel Binaural Silicon Cochlea for Event-Driven Stereo-Audio Sensing". In: *IEEE Journal of Solid-State Circuits* 51.11, pp. 2554–2569.

Yousefzadeh, Amirreza, Mirosław Jabłoński, Taras Iakymchuk, Alejandro Linares-Barranco, Alfredo Rosado, Luis A Plana, Steve Temple, Teresa Serrano-Gotarredona, Steve B Furber, and Bernabé Linares-Barranco (2017). "On multiple AER handshaking channels over high-speed bit-serial bidirectional LVDS links with flow-control and clock-correction on commercial FPGAs for scalable neuromorphic systems". In: *IEEE transactions on biomedical circuits and systems* 11.5, pp. 1133–1147.

Youssef, Ibrahim, Mehmet Mutlu, Behzad Bayat, Alessandro Crespi, Simon Hauser, Jörg Conradt, Alexandre Bernardino, and Auke Ijspeert (2020). "A neuro-inspired computational model for a visually guided robotic lamprey using frame and event based cameras". In: *IEEE Robotics and Automation Letters* 5.2, pp. 2395–2402.

Yu, Theodore, Andrew Schwartz, John Harris, Malcolm Slaney, and Shih-Chii Liu (2009). "Periodicity detection and localization using spike timing from the AER EAR". In: *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*. IEEE, pp. 109–112.

Zeng, Fan-Gang, Stephen Rebscher, William Harrison, Xiaoan Sun, and Haihong Feng (2008). "Cochlear implants: system design, integration, and evaluation". In: *IEEE reviews in biomedical engineering* 1, pp. 115–142.

Zhao, Jingyue et al. (2020). "Neuromorphic implementation of spiking relational neural network for motor control". In: *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, pp. 89–93.

Zilany, Muhammad SA and Ian C Bruce (2006). "Modeling auditory-nerve responses for high sound pressure levels in the normal and impaired auditory periphery". In: *The Journal of the Acoustical Society of America* 120.3, pp. 1446–1466.

Zwicker, Eberhard and Hugo Fastl (2013). *Psychoacoustics: Facts and models*. Vol. 22. Springer Science & Business Media.

Part II

Appendices

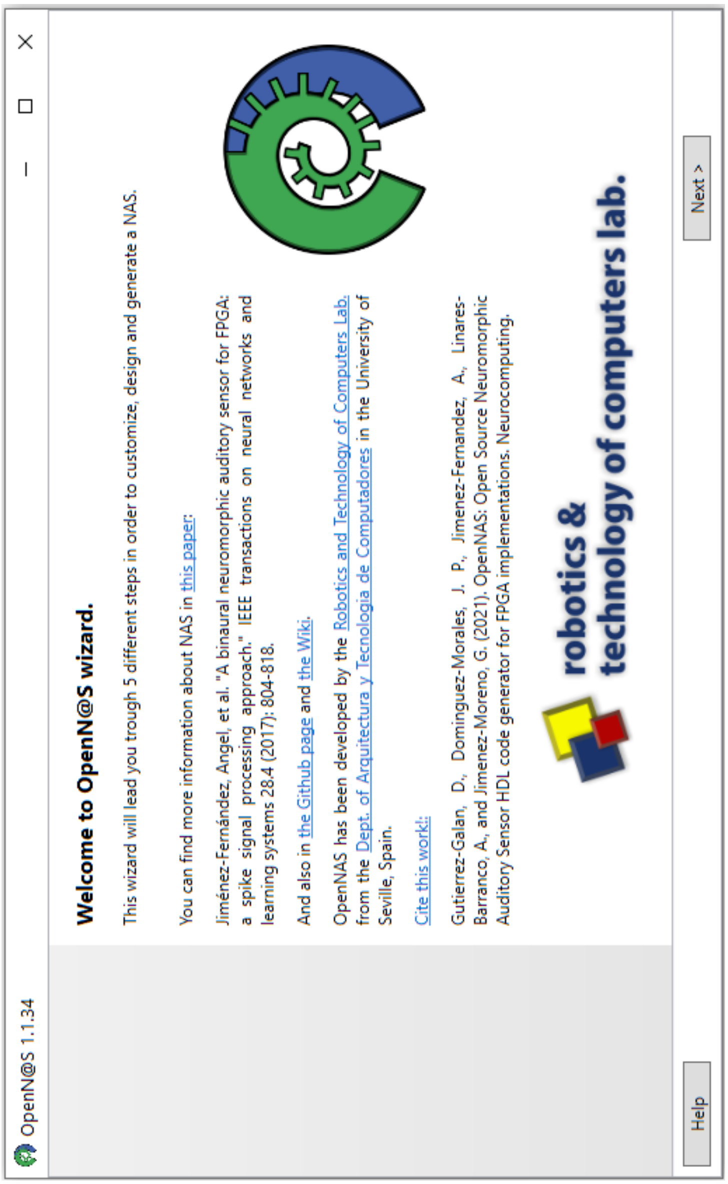
Appendix A

OpenNAS software tool

A.1 OpenNAS screens summary

1. Software metadata:
   - Current software version: v1.1.34.
   - Permanent link to executables of this version: https://github.com/RTC-research-group/OpenNAS/releases/tag/1.1.34.
   - Permanent link to code/repository used of this code version: http://github.com/RTC-research-group/OpenNAS.
   - Software License: GNU General Public License (GPL).
   - Computing platform/Operating System: Microsoft Windows.
   - Installation requirements & dependencies: Microsoft .NET Framework 4.5 or greater.
   - Software code languages, tools, and services used: C# and VHDL.
   - Link to user manual: http://github.com/RTC-research-group/OpenNAS/tree/master/OpenNAS/Wiki_files/User_manual/.
   - Link to developer documentation or manual: https://github.com/RTC-research-group/OpenNAS/tree/master/OpenNAS/SandCastleBuilder/Help.
   - Support email for questions: dgutierrez@atc.us.es.

A.1.1 OpenNAS welcome screen



Welcome to OpenN@S wizard.

This wizard will lead you trough 5 different steps in order to customize, design and generate a NAS.

You can find more information about NAS in this paper:

Jiménez-Fernández, Angel, et al. "A binaural neuromorphic auditory sensor for FPGA: a spike signal processing approach." IEEE transactions on neural networks and learning systems 28.4 (2017): 804-818.

And also in the Github page and the Wiki.

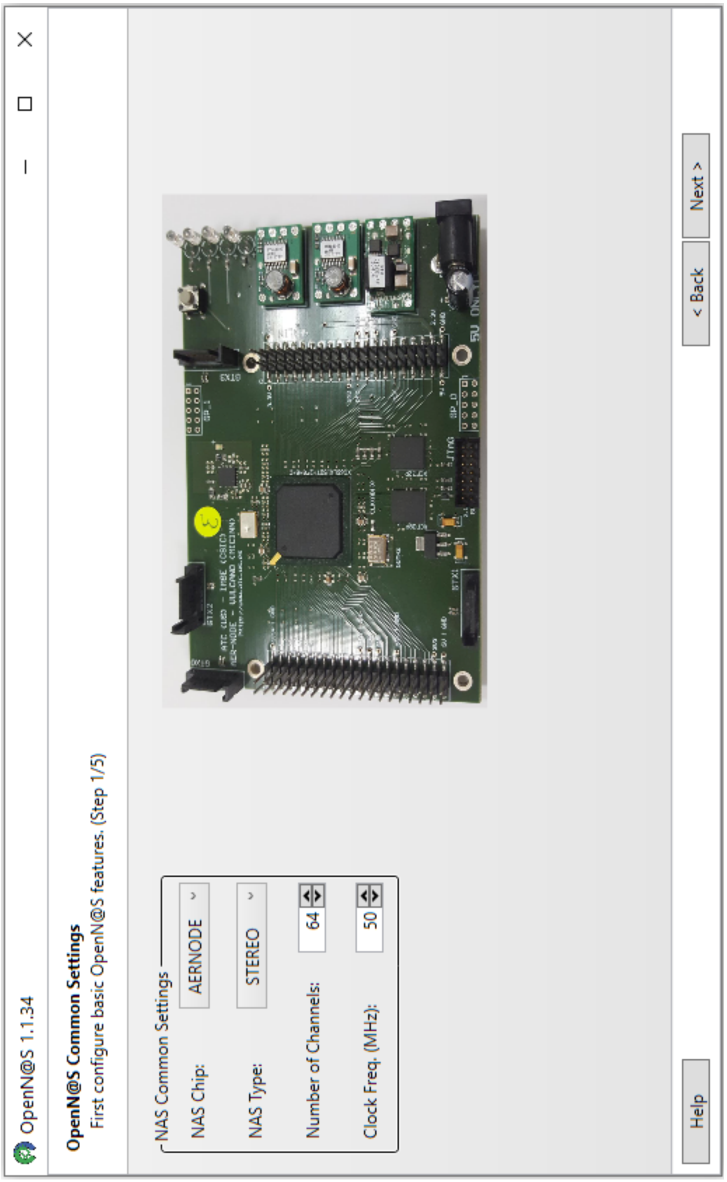
OpenNAS has been developed by the Robotics and Technology of Computers Lab. from the Dept. of Arquitectura y Tecnologia de Computadores in the University of Seville, Spain.

Cite this work!:

Gutierrez-Galan, D., Dominguez-Morales, J. P., Jimenez-Fernandez, A., Linares-Barranco, A., and Jimenez-Moreno, G. (2021). OpenNAS: Open Source Neuromorphic Auditory Sensor HDL code generator for FPGA implementations. Neurocomputing.

FIGURE A.1: OpenNAS tool welcome message.

A.1.2 OpenNAS common settings screen


OpenN@S 1.1.34

OpenN@S Common Settings

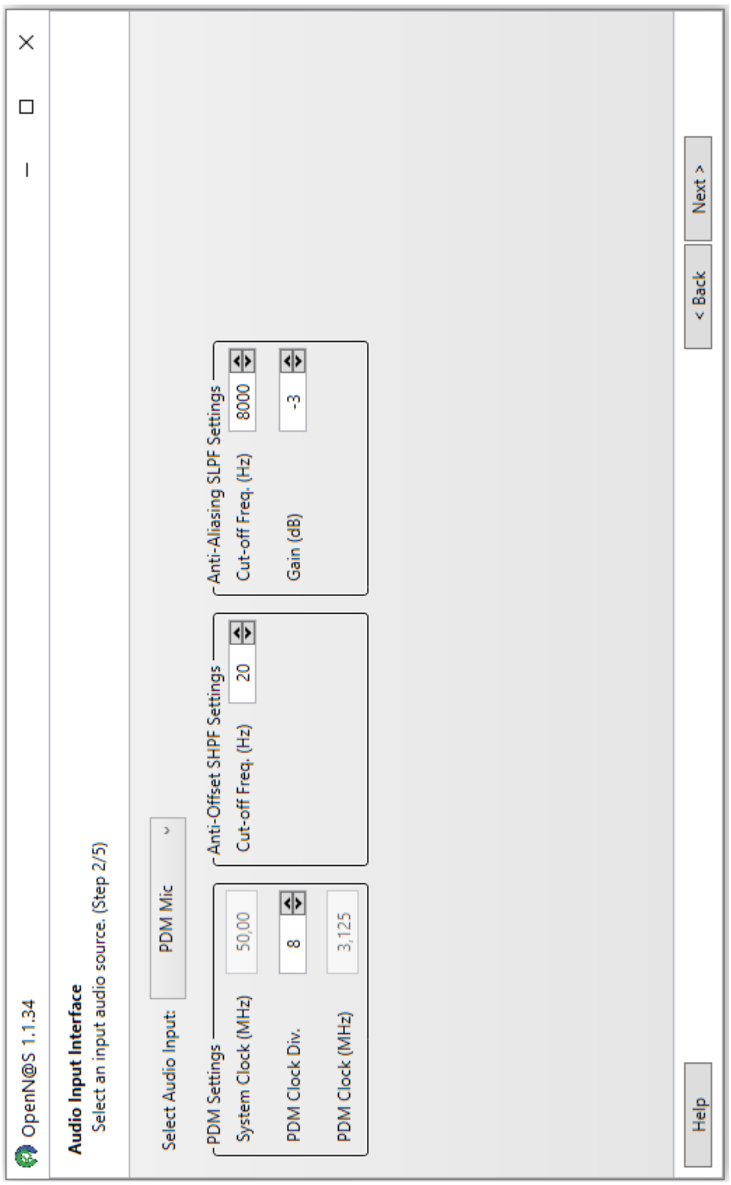
First configure basic OpenN@S features. (Step 1/5)

NAS Common Settings

NAS Chip:	AERNODE
NAS Type:	STEREO
Number of Channels:	64
Clock Freq. (MHz):	50

FIGURE A.2: OpenNAS screen for step 1: common settings.

A.1.3 OpenNAS input interface screen



OpenN@S 1.1.34

Audio Input Interface

Select an input audio source. (Step 2/5)

Select Audio Input: PDM Mic

PDM Settings

System Clock (MHz)	50,00
PDM Clock Div.	8
PDM Clock (MHz)	3,125

Anti-Offset SHPF Settings

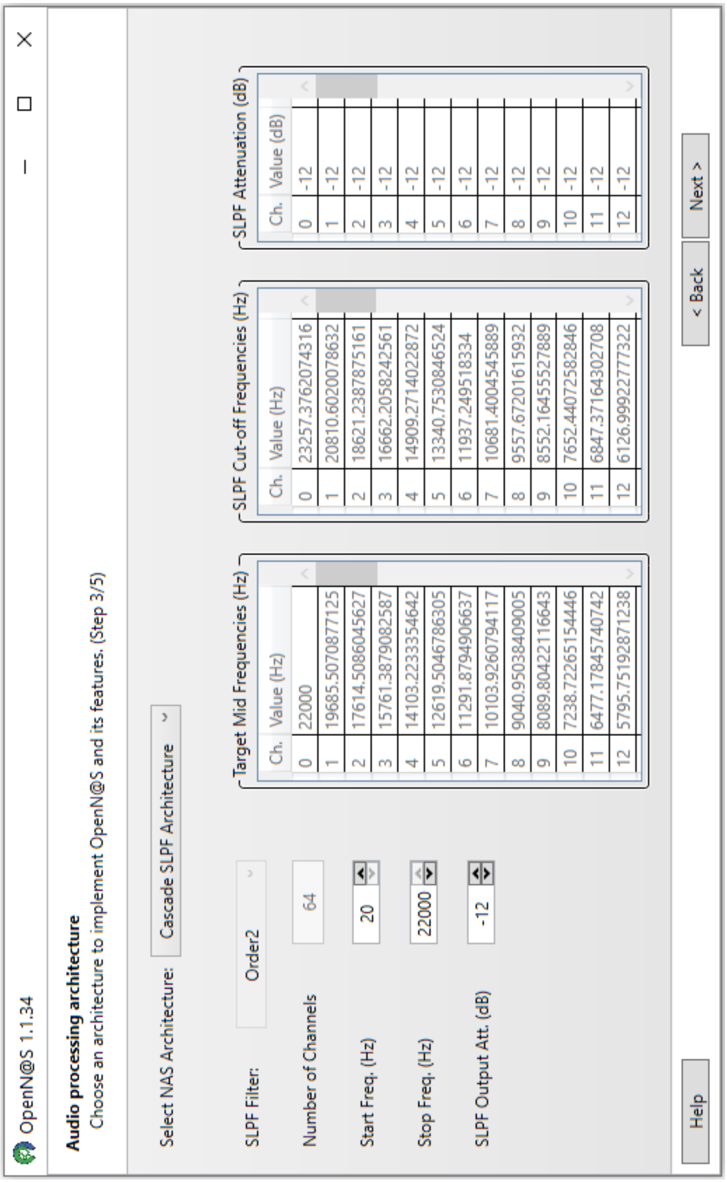
Cut-off Freq. (Hz)	20

Anti-Aliasing SLPF Settings

Cut-off Freq. (Hz)	8000
Gain (dB)	-3

FIGURE A.3: OpenNAS screen for step 2: audio input interface.

A.1.4 OpenNAS processing architecture screen



Audio processing architecture

Choose an architecture to implement OpenN@S and its features. (Step 3/5)

Select NAS Architecture: Cascade SLPF Architecture

SLPF Filter: Order2

Number of Channels: 64

Start Freq. (Hz): 20

Stop Freq. (Hz): 22000

SLPF Output Att. (dB): -12

Target Mid Frequencies (Hz)

Ch.	Value (Hz)
0	22000
1	19685.5070877125
2	17614.5086045627
3	15761.3879082587
4	14103.2233354642
5	12619.5046786305
6	11291.8794906637
7	10103.9260794117
8	9040.95038409005
9	8089.80422116643
10	7238.72265154446
11	6477.17845740742
12	5795.75192871238

SLPF Cut-off Frequencies (Hz)

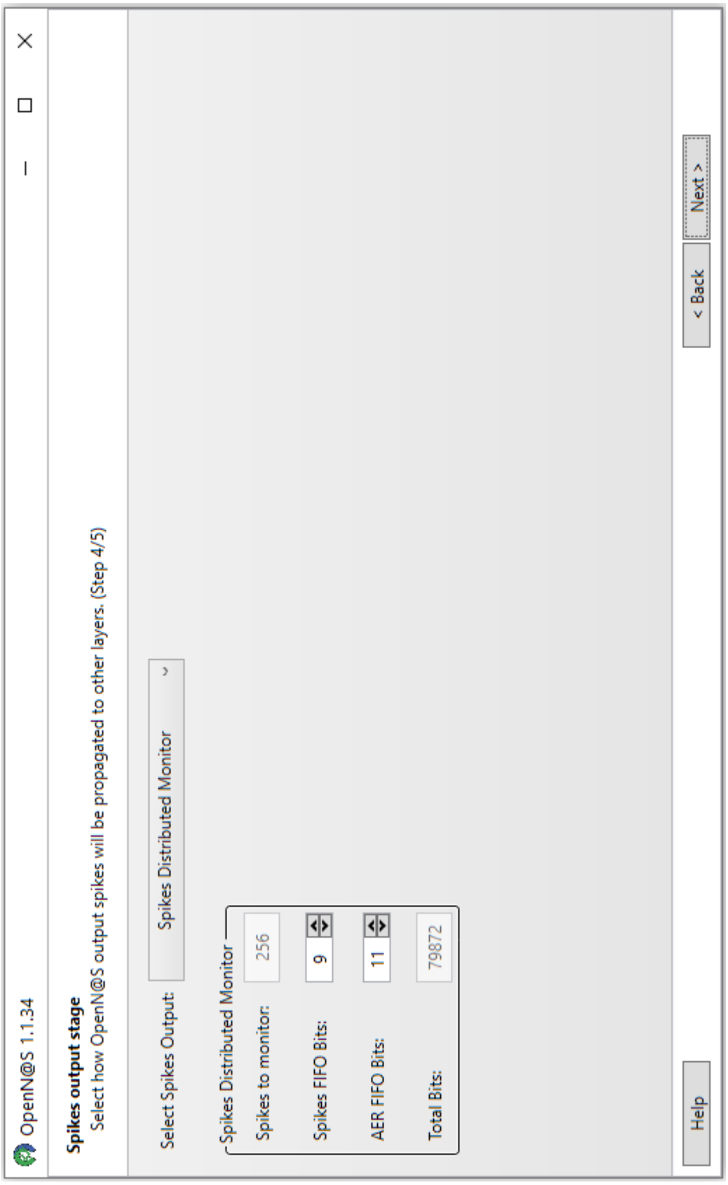
Ch.	Value (Hz)
0	23257.3762074316
1	20810.6020078632
2	18621.2387875161
3	16662.2058242561
4	14909.2714022872
5	13340.7530846524
6	11937.249518334
7	10681.4004545889
8	9557.67201615932
9	8552.16455527889
10	7652.44072582846
11	6847.37164302708
12	6126.99922777322

SLPF Attenuation (dB)

Ch.	Value (dB)
0	-12
1	-12
2	-12
3	-12
4	-12
5	-12
6	-12
7	-12
8	-12
9	-12
10	-12
11	-12
12	-12

FIGURE A.4: OpenNAS screen for step 3: audio processing architecture.

A.1.5 OpenNAS output interface screen


OpenN@S 1.1.34

Spikes output stage

Select how OpenN@S output spikes will be propagated to other layers. (Step 4/5)

Select Spikes Output: Spikes Distributed Monitor

Spikes Distributed Monitor

Spikes to monitor: 256

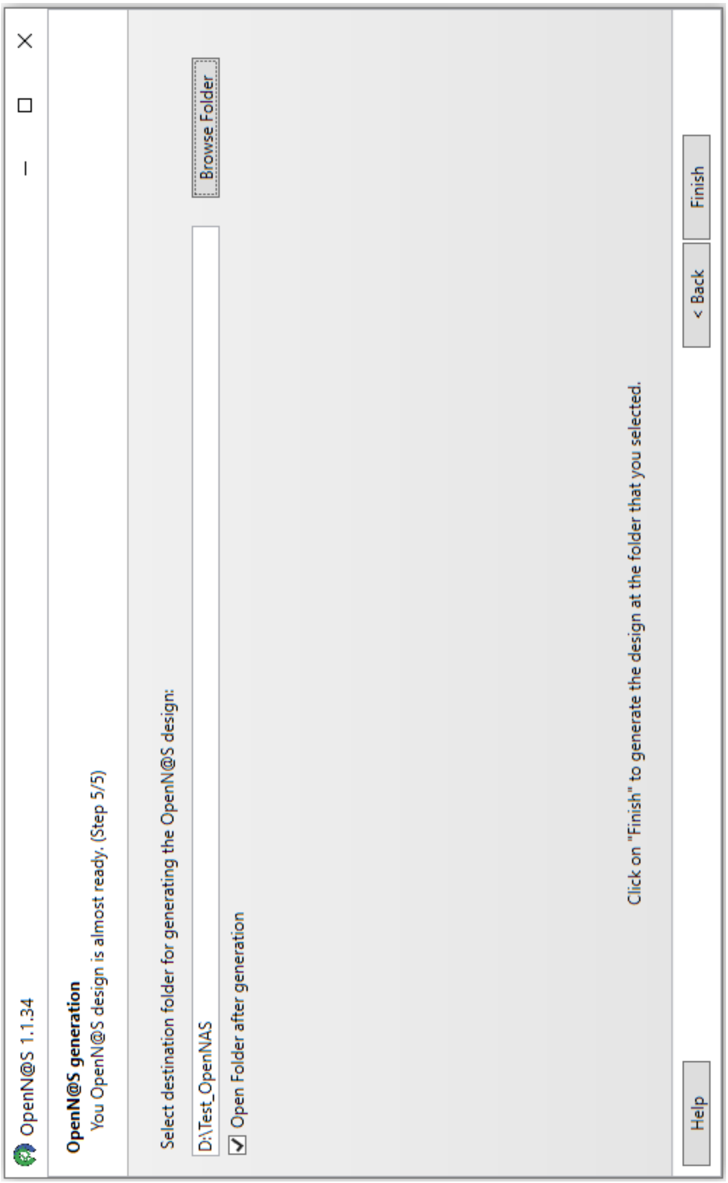
Spikes FIFO Bits: 9

AER FIFO Bits: 11

Total Bits: 79872

FIGURE A.5: OpenNAS screen for step 4: spiking output interface.

A.1.6 OpenNAS destination folder screen


OpenN@S 1.1.34

OpenN@S generation

You OpenN@S design is almost ready. (Step 5/5)

Select destination folder for generating the OpenN@S design:

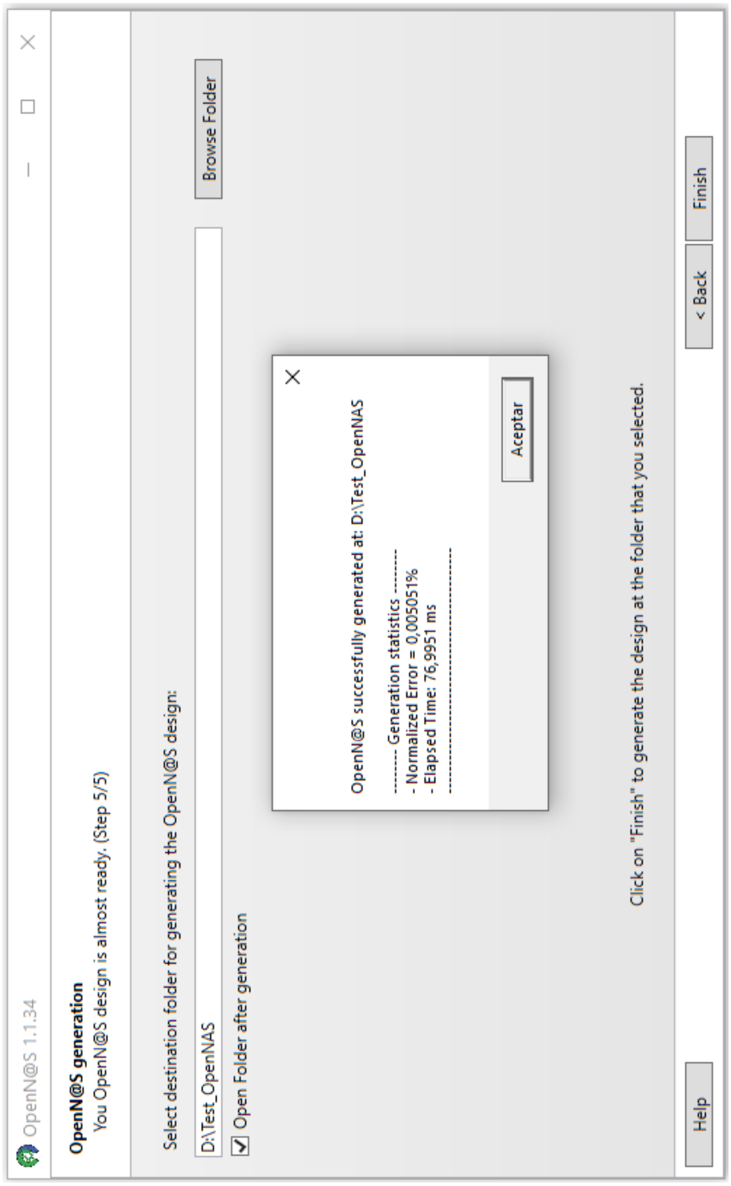
D:\Test_OpenNAS

[✓] Open Folder after generation

Click on "Finish" to generate the design at the folder that you selected.

FIGURE A.6: OpenNAS screen for step 5: destination folder selection.

A.1.7 OpenNAS generation success screen


OpenN@S 1.1.34

OpenN@S generation

You OpenN@S design is almost ready. (Step 5/5)

Select destination folder for generating the OpenN@S design:

D:\Test_OpenNAS

✓ Open Folder after generation

```
OpenN@S successfully generated at: D:\Test_OpenNAS

-------- Generation statistics --------
- Normalized Error = 0,005051%
- Elapsed Time: 76,9951 ms
---------------------------------------
```

Click on "Finish" to generate the design at the folder that you selected.

FIGURE A.7: OpenNAS final screen: code generator statistics.

Appendix B

NASIC test PCB

B.1 NASIC test PCB files

1. Features:
   - 2 PDM microphones interface.
   - 1 CS5343 audio ADC with I2S interface.
   - 1 3.5mm audio jack female connector.
   - 1 generic purpose jumper.
   - 10-pin connector for FPGA interface.
   - 3.3V power supply.
2. Hardware requirements:
   - Any FPGA- or microcontoller-based board with I2S interface.

B.1.1 Schematic: *NASIC_test_pcb.SchDoc*

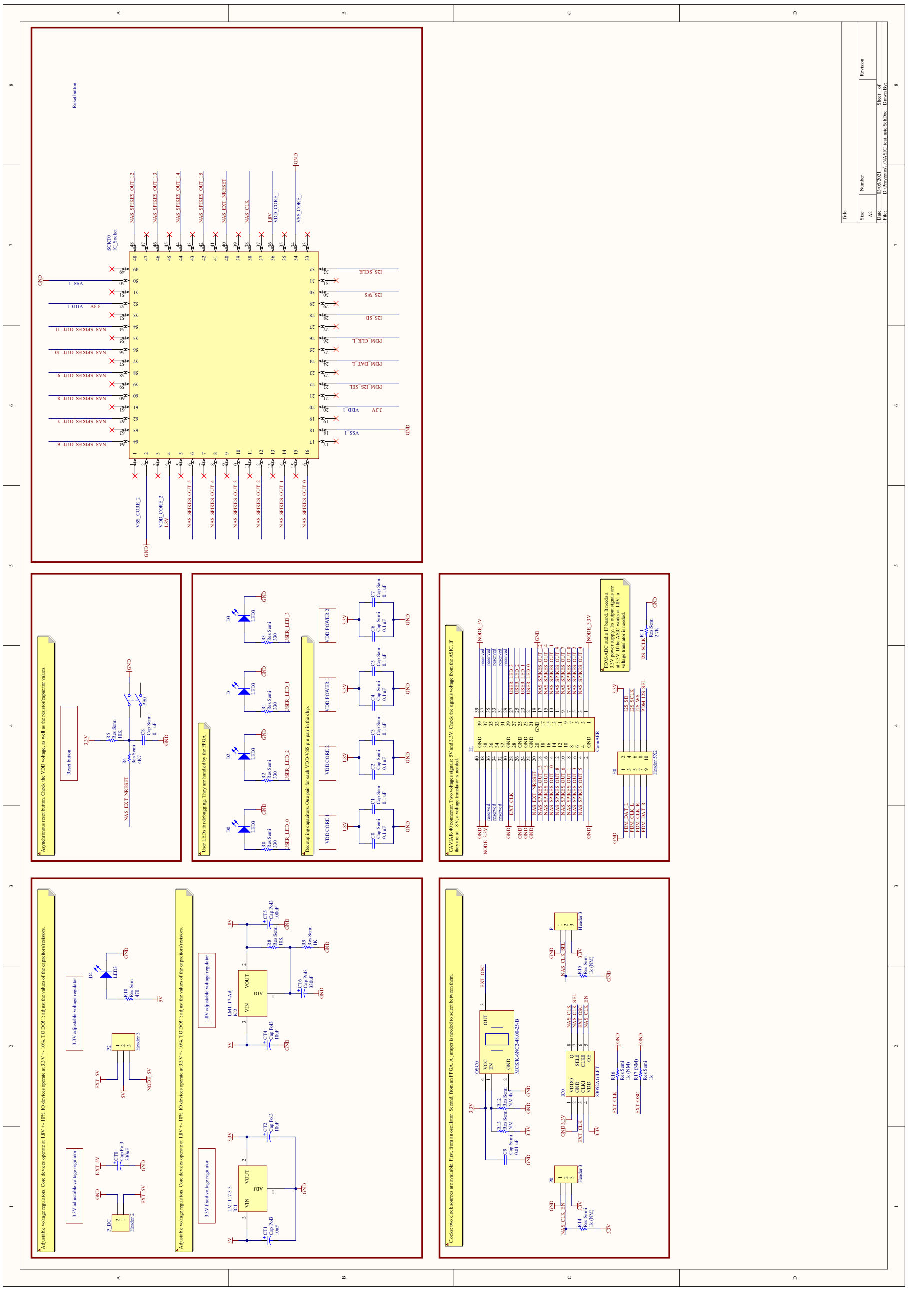


FIGURE B.1: NASIC test PCB schematic.

B.1.2 Board top: *NASIC_test_pcb.PcbDoc*

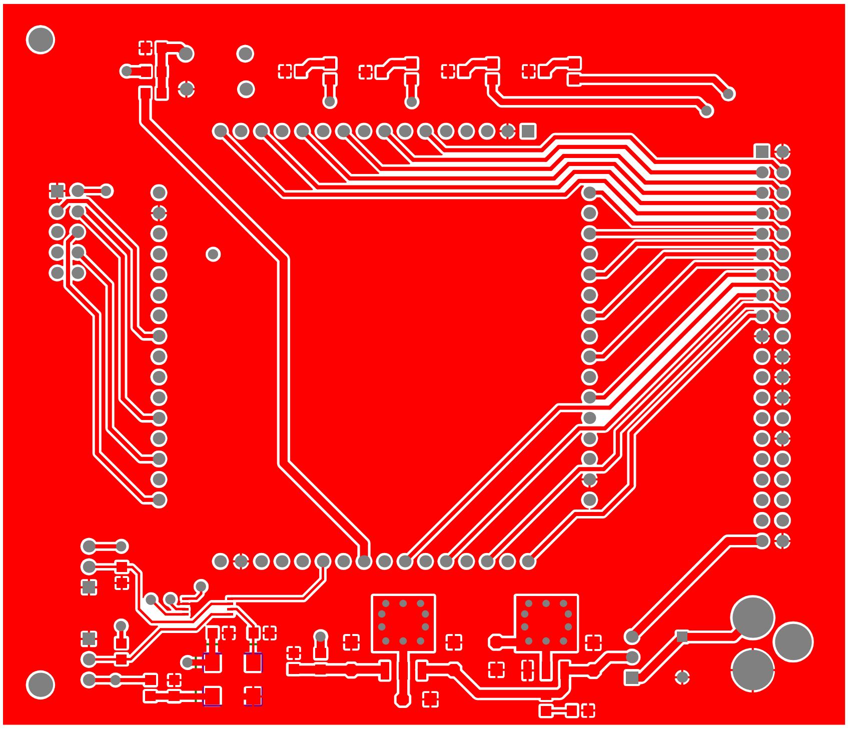


FIGURE B.2: NASIC test PCB board top view.

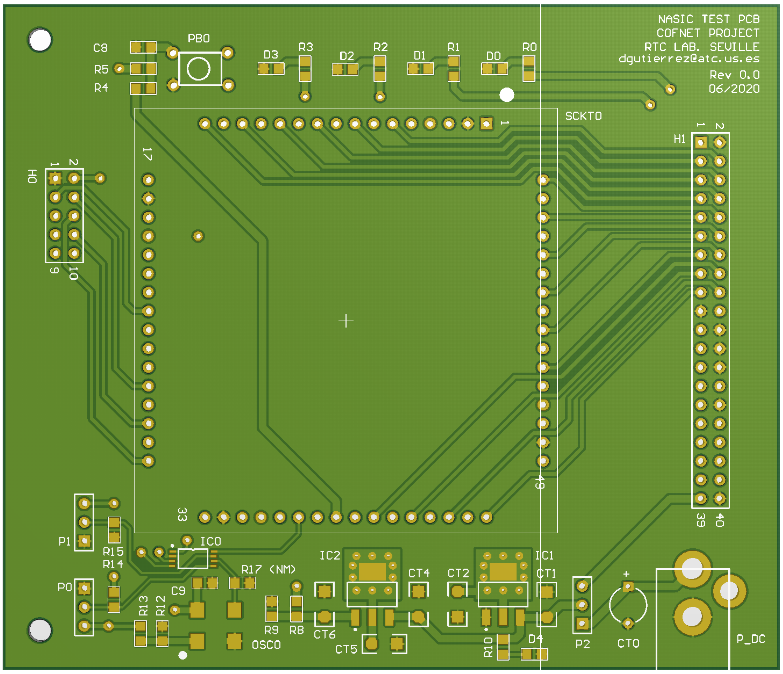


FIGURE B.3: NASIC test PCB board top 3D view.

B.1.3 Board bottom: *NASIC_test_pcb.PcbDoc*

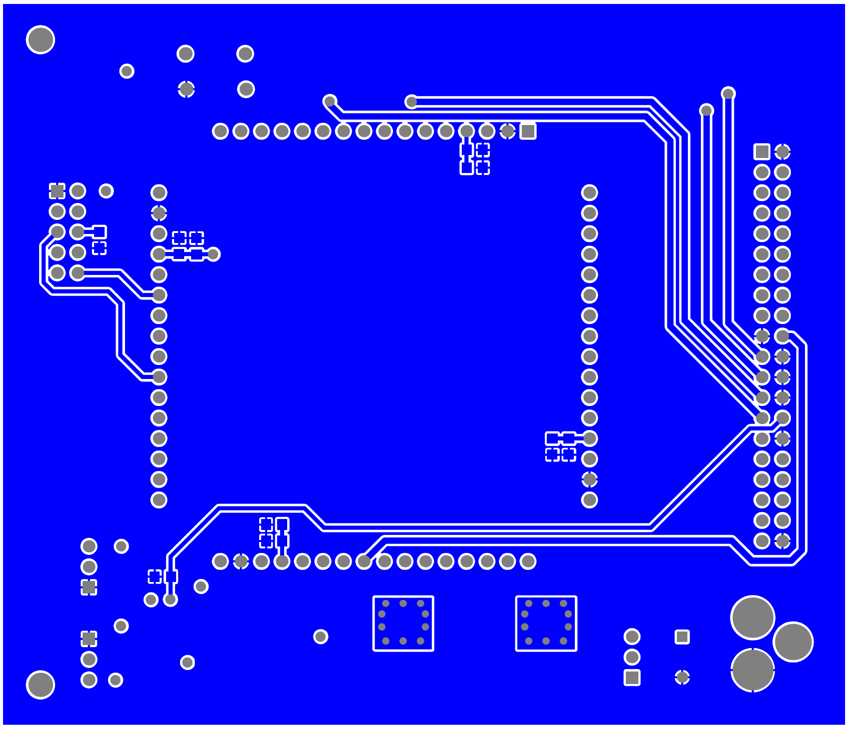


FIGURE B.4: NASIC test PCB board bottom view.

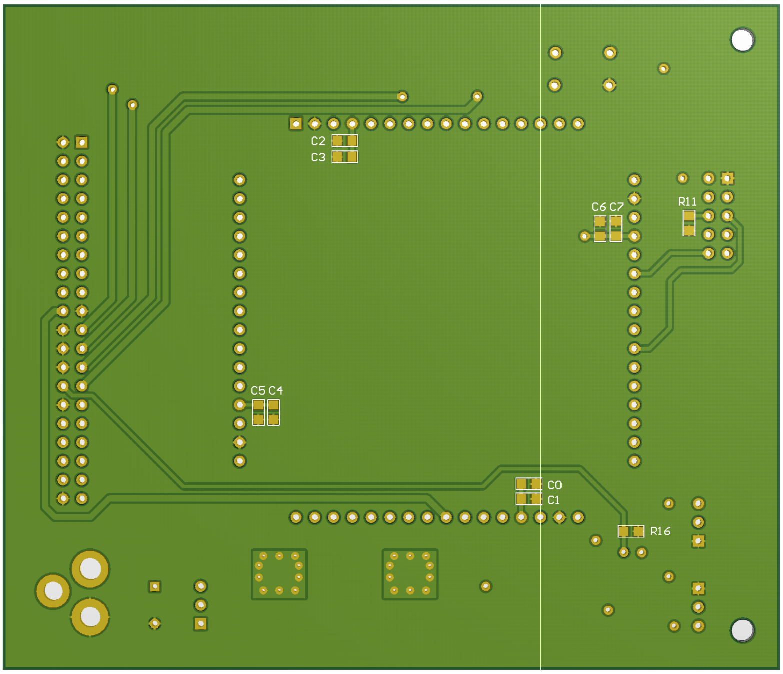


FIGURE B.5: NASIC test PCB board bottom 3D view.

Appendix C

Generic purpose PCBs

C.1 ADC-PDM microphones board PCB files

1. Features:
   - 2 PDM microphones interface.
   - 1 CS5343 audio ADC with I2S interface.
   - 1 3.5mm audio jack female connector.
   - 1 generic purpose jumper.
   - 10-pin connector for FPGA interface.
   - 3.3V power supply.
2. Hardware requirements:
   - Any FPGA- or microcontoller-based board with I2S interface.

C.1.1 Schematic: *ADC_PDM_mic_board.SchDoc*

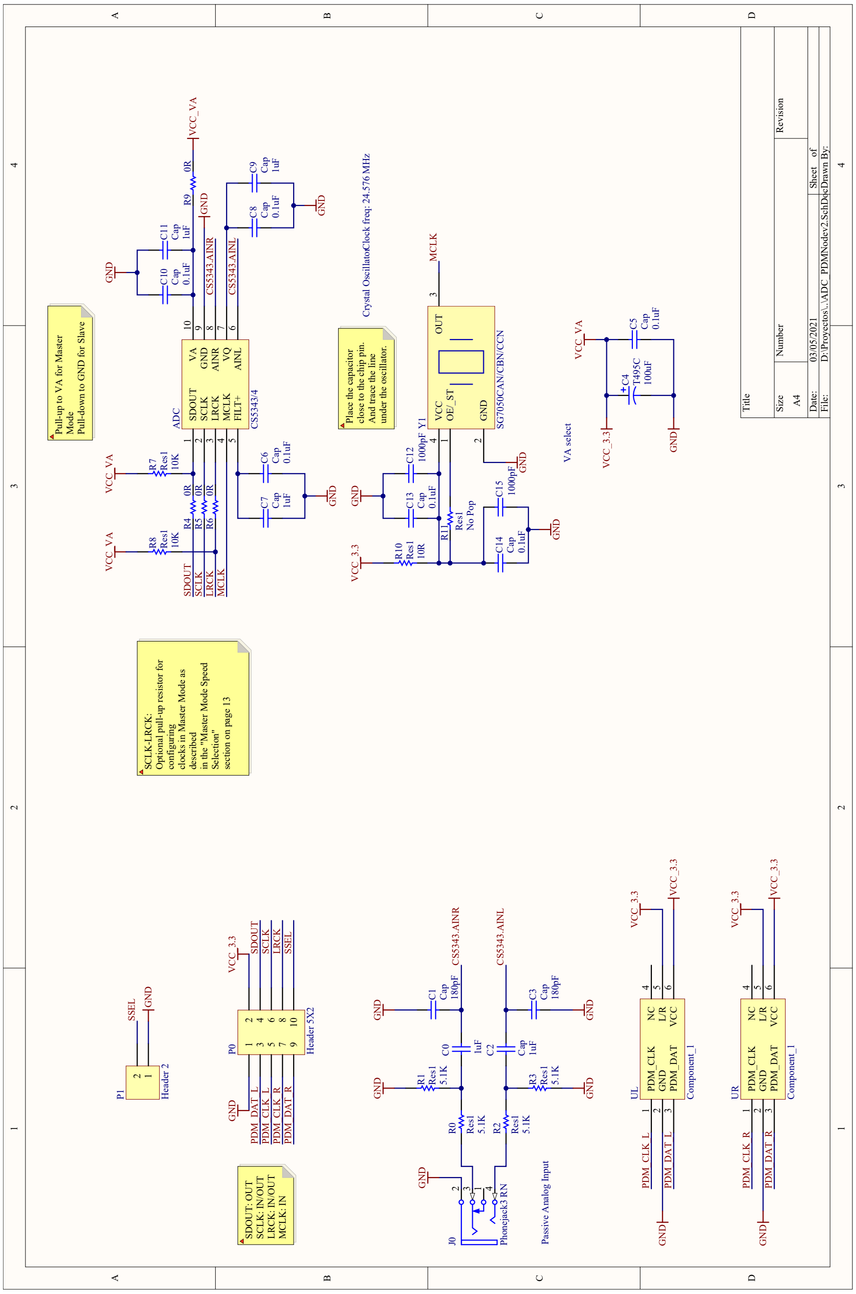

Pull-up to VA for Master Mode
Pull-down to GND for Slave
Place the capacitor close to the chip pin. And trace the line under the oscillator.
Crystal OscillatorClock freq: 24.576 MHz
SCLK-LRCK: Optional pull-up resistor for configuring clocks in Master Mode as described in the "Master Mode Speed Selection" section on page 13
SDOUT: OUT
SCLK: IN/OUT
LRCK: IN/OUT
MCLK: IN
ADC
CS5343/4
SG7050CAN/CBN/CCN
VA select
Passive Analog Input
Phonejack3 RN
Header 5X2
Header 2
Component_1
VCC_VA
VCC_3.3
GND
MCLK
Title
Size A4
Number
Revision
Date: 03/05/2021
File: D:\Proyectos\..\ADC_PDMNodev2.SchDoc
Sheet of
Drawn By:

FIGURE C.1: ADC-PDM microphones board PCB schematic.

C.1.2 Board top: *ADC_PDM_mic_board.PcbDoc*

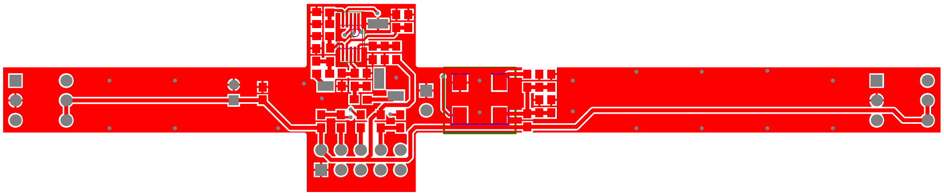


FIGURE C.2: ADC-PDM microphones board PCB board top view.

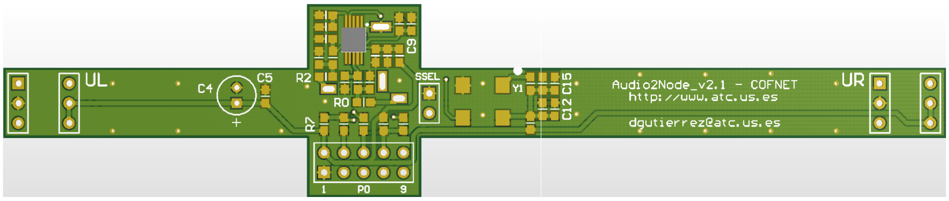


FIGURE C.3: ADC-PDM microphones board PCB board top 3D view.

C.1.3 Board bottom: *ADC_PDM_mic_board.PcbDoc*

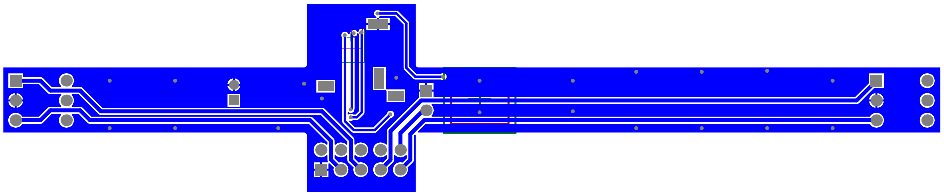


FIGURE C.4: ADC-PDM microphones board PCB board bottom view.

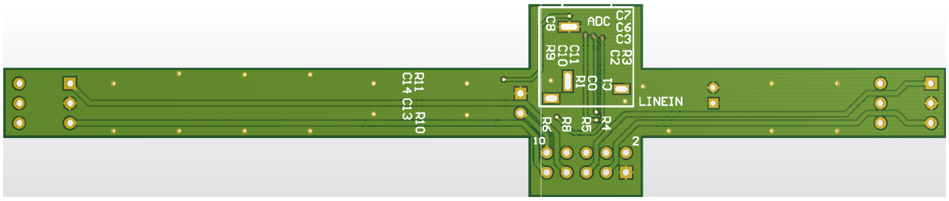


FIGURE C.5: ADC-PDM microphones board PCB board bottom 3D view.

C.2 ZTEX 2.13 base board PCB files

1. Features:
   - 8 general purpose LEDs.
   - 4 general purpose buttons.
   - 2 general purpose jumpers.
   - SpiNNaker interface through SpiNN-link connector.
   - ADC-PDM microphones board interface (see C.1).
   - CAVIAR-20 connector with 20 I/O pins.
   - JTAG connector for FPGA programming.
2. Hardware requirements:
   - ZTEX 2.13 board: from `https://www.ztex.de/usb-fpga-2/usb-fpga-2.13.e.html`.

C.2.1 Schematic: *ZTEX_base_board.SchDoc*

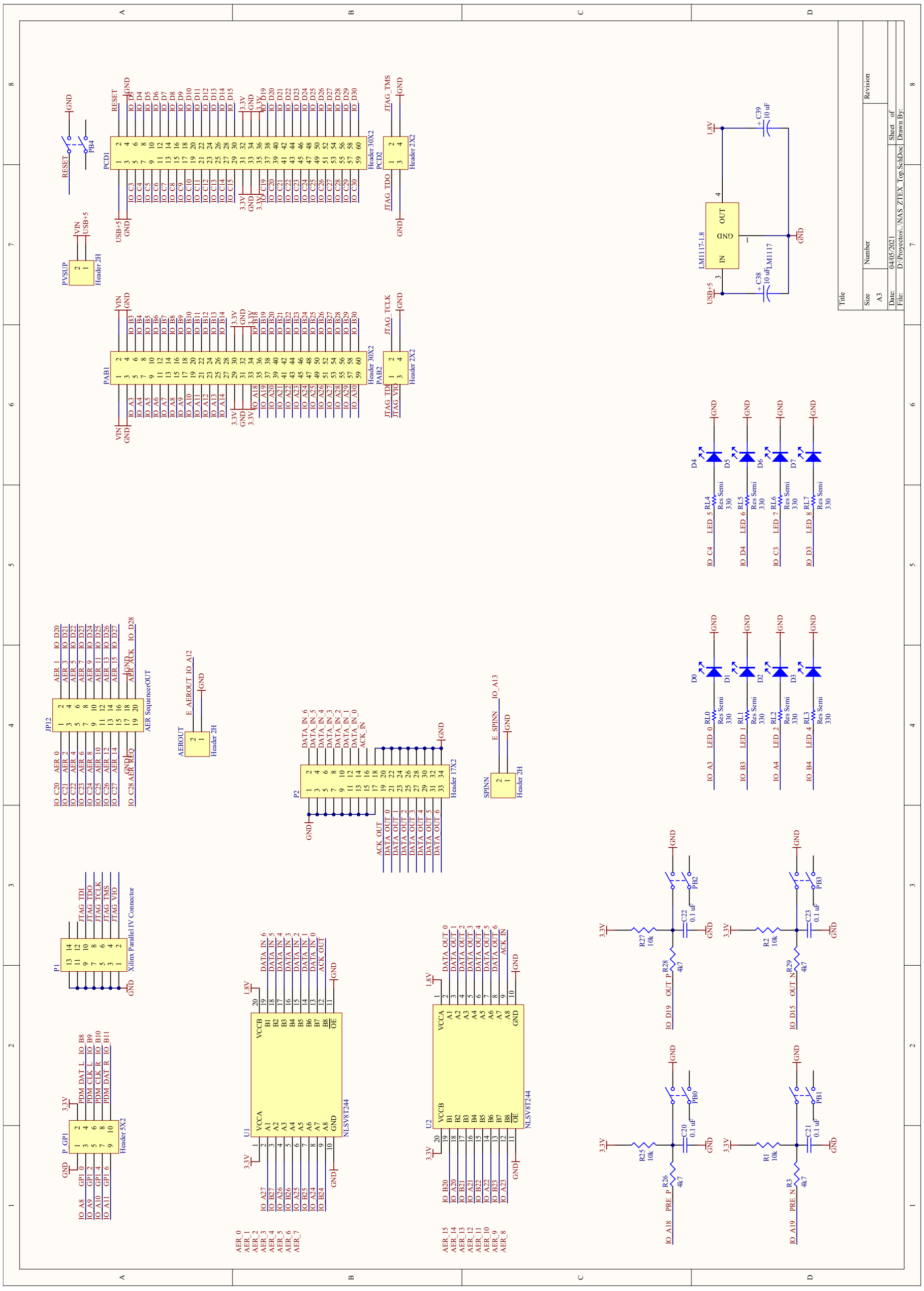


FIGURE C.6: ZTEX 2.13 base board PCB schematic.

C.2.2 Board top: *ZTEX_base_board.PcbDoc*

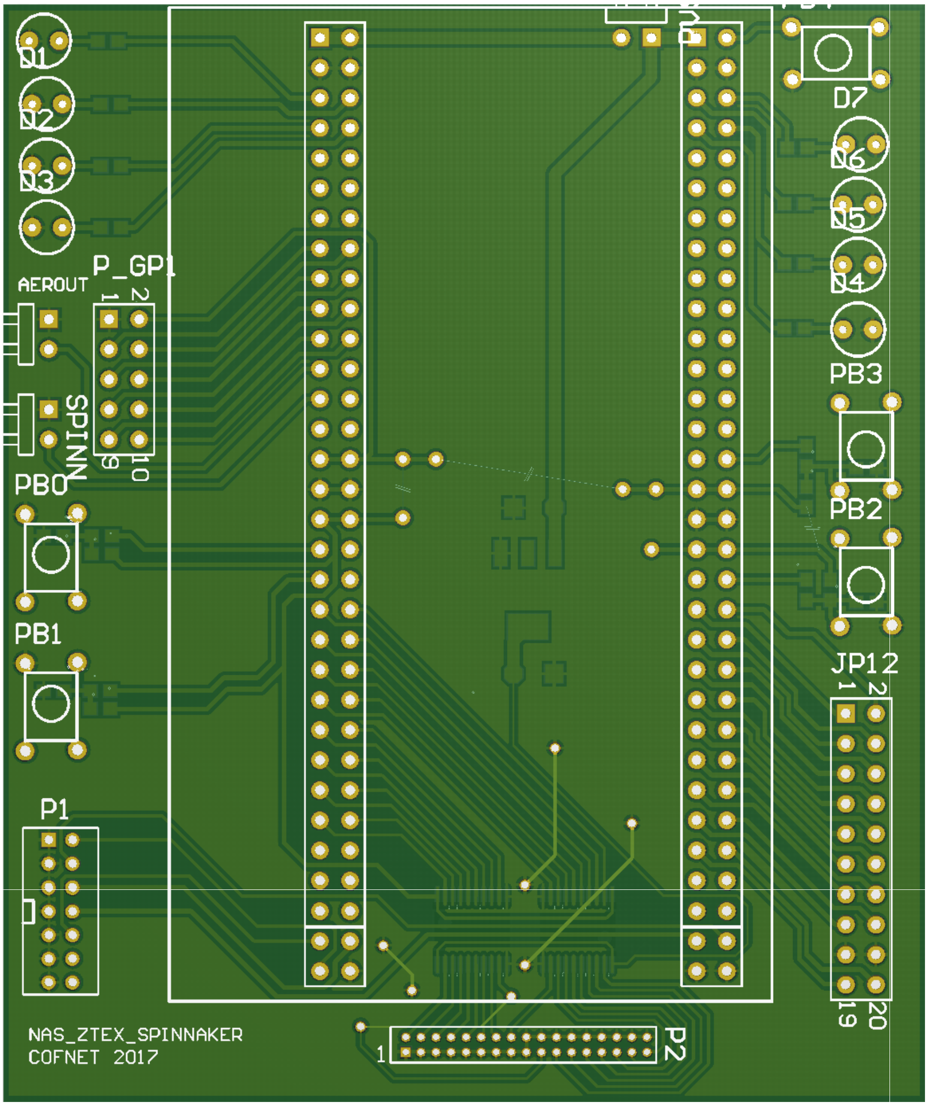


FIGURE C.7: ZTEX 2.13 base board top view.

C.2.3 Board bottom: *ZTEX_base_board.PcbDoc*

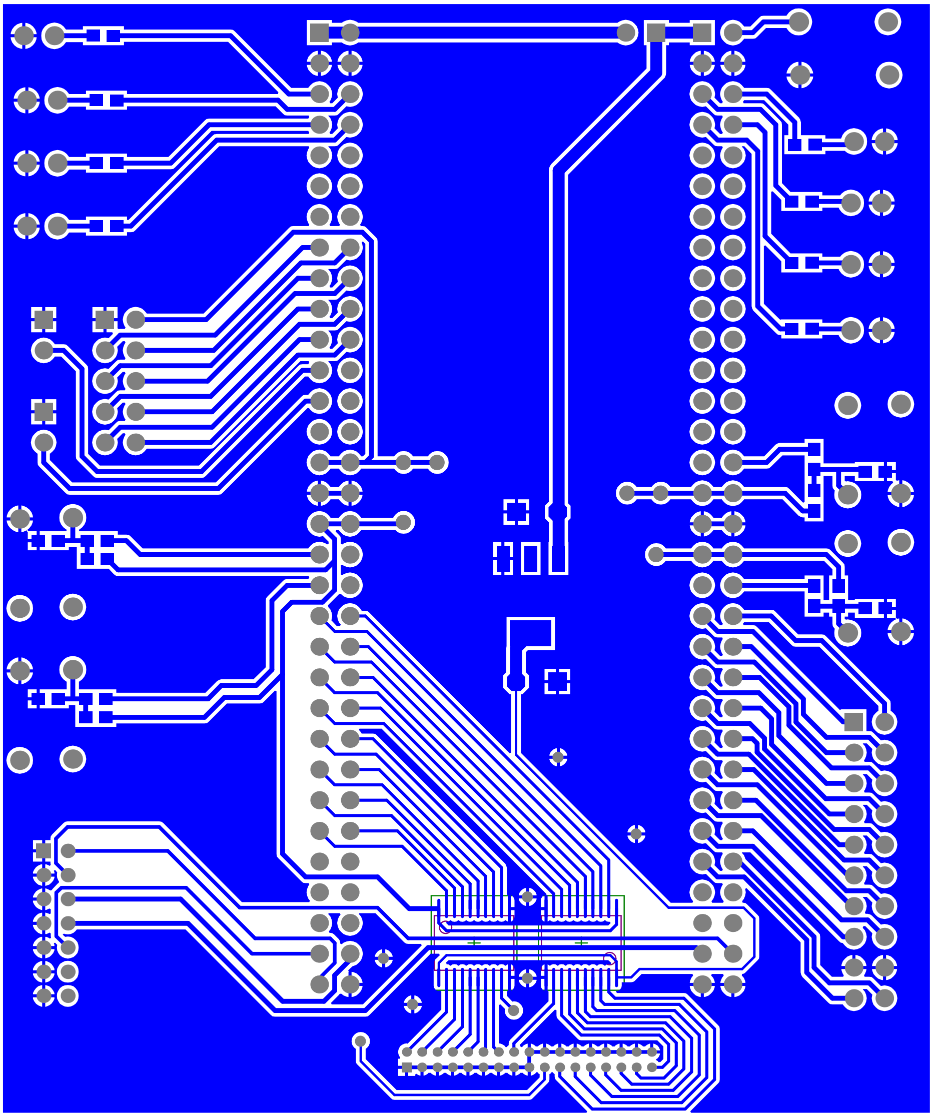


FIGURE C.8: ZTEX 2.13 base board bottom view.

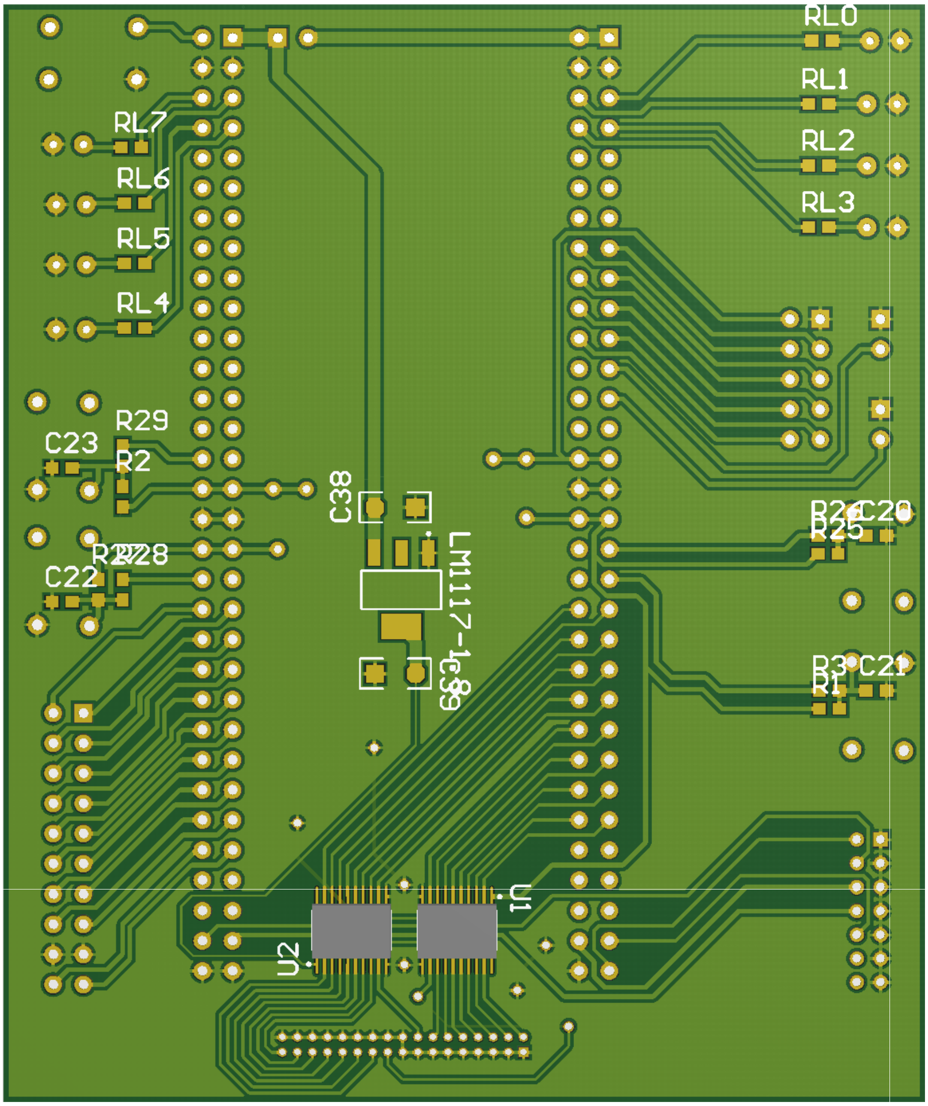


FIGURE C.9: ZTEX 2.13 base board bottom 3D view.