DISS. ETH NO. 27602

HARDWARE SYSTEMS FOR LOW-LATENCY AUDIO PROCESSING: EVENT-BASED AND MULTICHANNEL SYNCHRONOUS SAMPLING APPROACHES

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH

(Dr. sc. ETH Zurich)

presented by

ILYA KISELEV

M.Sc. in Applied Mathematics and Physics,
Moscow Institute of Physics and Technology

born on 08.09.1978

citizen of Russia

accepted on the recommendation of

Prof. Dr. Shih-Chii Liu
Prof. Dr. Richard Hahnloser
Prof. Dr. Jörg Conradt

2021

ABSTRACT

Neuromorphic technology is slowly maturing with a variety of useable event-driven spiking sensors and hardware implementations of spiking neural networks. Sensory processing algorithms are still under investigation and their usefulness in natural environments are still relatively unexplored compared to algorithms using conventional sensors and digital hardware.

We developed hardware test beds that allow us to explore event-based sensory processing algorithms and regular sampling based algorithms in real-world conditions. The goal of my thesis is three-fold: 1) to develop a hardware test bed for implementing spiking networks together with spiking sensors to study a possibility of using multiple sensors of different modalities to improve classification performance in real-world conditions; 2) to implement a local automatic gain control mechanism to increase the input dynamic range of a spiking cochlea operating in natural environments where the sound dynamic range can be greater than 60 dB; 3) to implement a multi-microphone hardware platform that can be used for real-time beamforming as part of a wireless acoustic sensor network.

The first part of the thesis describes development of a real-time hardware system that fuses information from neuromorphic spiking sensors of different modalities. The core of the system is a general purpose accelerator for spiking Deep Neural Networks (DNN) implemented on a Field-Programmable Gate Array (FPGA). We demonstrate the performance of the system on an audio-visual sensor fusion task using a Dynamic Vision Sensor (DVS) and a Dynamic Audio Sensor (DAS) spiking sensors for classification of digits from the Modified National Institute of Standards and Technology (MNIST) dataset augmented with specific audio tones for each digit.

We demonstrate that reliable classification is possible with just a fraction of spikes produced by the sensors. On the other hand, processing the full stream of spikes increases the computational demand of the system proportionally to the increase of the spike rate. In addition, the spike rate of the audio sensor depends on the input signal amplitude, which makes it difficult to train classifiers to be invariant to input signals with a wide dynamic range. However, it is known that biological audio and visual processing systems can accommodate to input signals that differ by orders of magnitude, while maintaining a moderate neuron spike rate.

The second part of the thesis addresses the problem of increasing spike rates in response to high amplitude signals in the spiking silicon cochlea by developing a local spike-based gain control algorithm, that constantly monitors the spike rate at the output of each channel and adapts the corresponding channel gain, so that its spike rate would not exceed a predefined threshold. We implemented this algorithm in hardware for the Dynamic Audio Sensor Low Power (DASLP) silicon cochlea and studied its performance on synthetic tests and real audio classification problem.

The third part of the thesis work is carried out within a multi-partner European project, COCOHA (COgnitive COntrol of a Hearing Aid, www.cocoha.org), that

aimed to develop a system for attention decoding from electroencephalogram (EEG) signals for directing the speech of an attended talker to the user of a hearing aid device. The goal of this work is to construct a synchronized distributed multi-microphone platform which can be used for general auditory scene analysis. The developed platform is composed of multi-microphone modules which can perform synchronized audio sampling at different parts of the room and transmit the audio streams with low latency to a central processing unit, where the samples from different microphones can be aligned with a sub-microsecond precision. Synchronized sampling across the ad-hoc distributed microphone array enables a variety of algorithms to be used for further processing, e.g. for tasks such as beamforming, source separation or speech enhancement. The platform was used for testing a set of beamforming algorithms in the wild.

All three parts serve a common goal of enabling application of novel auditory sensing technology in practically relevant settings, by coping with challenges of real-world deployment.

ZUSAMMENFASSUNG

Die neuromorphe Technologie reift stetig mit einer Vielzahl anwendbarer ereignisgesteuerter Spike-Sensoren und Hardware-Implementierungen von gepulsten neuronalen Netzen (engl. Spiking Neural Networks). Sensorische Verarbeitungsalgorithmen werden derzeit untersucht und ihre Nützlichkeit in natürlichen Umgebungen ist im Vergleich zu Algorithmen mit konventionellen Sensoren und digitaler Hardware noch verhältnismässig unerforscht. Wir haben Hardware-Prüfstände entwickelt, mit denen wir ereignisbasierte sensorische Verarbeitungsalgorithmen und reguläre samplingbasierte Algorithmen unter realen Bedingungen untersuchen können. In meiner Thesis verfolge ich drei Ziele: 1) Die Entwicklung eines Hardware-Prüfstands zur Implementierung von Spiking-Netzwerken zusammen mit Spiking-Sensoren, um die Möglichkeit zu untersuchen, mehrere Sensoren mit unterschiedlichen Modalitäten zu verwenden, um damit die Klassifizierungsleistung unter realen Bedingungen zu verbessern; 2) Implementierung eines lokalen automatischen Verstärkungsregelungsmechanismus zur Erhöhung des Eingangsdynamikbereichs einer Spike-Cochlea, die in natürlichen Umgebungen arbeitet, in denen der Schalldynamikbereich größer als 60 dB sein kann; 3) Implementierung einer Hardwareplattform mit mehreren Mikrofonen, die als als Teil eines drahtlosen akustischen Sensornetzwerks zur Echtzeit-Strahlformung verwendet werden kann.

Der erste Teil der Arbeit beschreibt die Entwicklung eines Echtzeit-Hardwaresystems, das Informationen von neuromorphen Spike-Sensoren unterschiedlicher Modalitäten zusammenführt. Der Kern des Systems ist ein Allzweckbeschleuniger für tiefe neuronale Netze (engl. Deep Neural Networks), welcher auf einem FPGA (engl. Field Programmable Gate Array) implementiert ist. Wir demonstrieren die Leistung des Systems anhand einer audiovisuellen Sensorfusionsaufgabe eines Sensors unter Verwendung eines DVS- (engl. Dynamic Vision Sensor) und eines DAS- (engl. Dynamic Audio Sensor) Spike-Sensors zur Klassifizierung von Ziffern aus dem Datensatz des Modified National Institute of Standards and Technology (MNIST), ergänzt um spezifische Audiotöne für jede Ziffer. Wir zeigen, dass eine zuverlässige Klassifizierung mit nur einem Bruchteil der von den Sensoren erzeugten Pulsen (Spikes) möglich ist. Andererseits erhöht die Verarbeitung der gesamten Pulsraten den Rechenaufwand des Systems proportional zur Erhöhung der Spitzenrate. Darüber hinaus hängen die Spitzenraten des Audiosensors von der Amplitude des Eingangssignals ab, was es schwierig macht, Klassifizierer so zu trainieren, dass sie für Eingangssignale mit großem Dynamikbereich geeignet sind. Es ist jedoch bekannt, dass biologische Audio- und visuelle Verarbeitungssysteme Eingangssignale verarbeiten können, die sich um Größenordnungen unterscheiden, während eine moderate Neuronenspitzenrate beibehalten wird. Der zweite Teil der Arbeit befasst sich mit dem Problem der Erhöhung der Spikeraten in Reaktion auf Signale mit hoher Amplitude in der Spiking Silicon Cochlea durch die Entwicklung eines lokalen Algorithmus zur Regelung der Verstärkung auf Spike-Basis, der die Spike-Rate am Ausgang jedes Kanals ständig überwacht und die entsprechende Ka-

nalverstärkung anpasst, so dass ihre Spikerate einen vordefinierten Schwellenwert nicht überschreitet. Wir haben diesen Algorithmus für die DASLP (engl. Dynamic Audio Sensor Low Power) Silicon Cochlea auf Hardwarebasis implementiert und seine Leistung anhand von snytethischen Tests und realen Audioklassifizierungsproblemen untersucht.

Der dritte Teil der Thesis wird im Rahmen eines europäischen Multi-Partner-Projekts, COCOHA (COgnitive COntrol of a Hearing Aid, www.cocoha.org), durchgeführt, mit dem Zweck ein System zur Aufmerksamkeitsdekodierung aus Elektroenzephalogrammsignalen (EEG) zu entwickeln, um die Sprache eines Redner, auf welchen die Aufmerksamkeit gerichtet ist, an den Benutzer eines Hörgerätes zu lenken. Das Ziel dieses Projekts ist es, eine synchronisierte, verteilte Multi-Mikrofon-Plattform aufzubauen, die für die allgemeine Analyse von Hörszenen verwendet werden kann. Die Plattform muss Multimikrofonmodule unterstützen, die synchronisiertes Audio-Sampling an verschiedenen Stellen des Raums durchführen und die Audio-Streams mit geringer Latenz an eine Zentraleinheit übertragen können, wo die Samples von verschiedenen Mikrofonen mit einer Genauigkeit von weniger als einer Mikrosekunde ausgerichtet werden können. Die synchronisierte Abtastung über das ad-hoc-verteilte Mikrofonarray, ermöglicht die Verwendung einer Vielzahl von Algorithmen zur weiteren Verarbeitung, z.B. für Aufgaben wie Strahlformung (engl Beamforming), Quellentrennung oder Sprachverbesserung. Die Plattform wurde zum Testen einer Reihe von Beamforming-Algorithmen in freier Wildbahn verwendet.

ACKNOWLEDGMENTS

Having done this work and looking back at the path I walked through I realize that it would not be possible without endless support and considerable guidance of my supervisor Prof. Dr. Shih-Chii Liu. I am boundlessly grateful to her for giving me an opportunity to accomplish this work, for her encouragement during the times when I was about to give up, and for her limitless patience and understanding. I am also grateful to my co-supervisor Prof. Dr. Richard Hanloser and co-examiner Prof. Dr. Jörg Conradt for their time and efforts.

I owe special thanks to my co-authors Daniel Neil and Enea Ceolini. Daniel designed and implemented the Minitaur spiking neural network accelerator, which was an inspiration and became a prototype for the new n-Minitaur system. He also provided the trained neural networks for the MNIST classification task. Enea implemented the software pipeline for data collection from WHISPER modules, packet level synchronization algorithm and all beamforming algorithms running on the WHISPER platform.

I appreciate contributions of Matthew Rahtz, who conducted latency measurements of Wi-Fi network for the WHISPER platform, and Chen-Han Chien, who implemented the code for DWM1000 data transfer testing. I thank Dr. Minhao Yang for answering all my questions about the DASLP cochlea chip, and Luca Longinotti for guiding me through his FPGA code for the DASLP board.

I was happy to be a member of the Sensors group, co-led by Prof. Dr. Tobi Delbruck, who could always give valuable feedback on my ideas and inspired me with his fascinating projects. I also appreciate his support with disentangling convoluted dependencies in the jAER software.

It was a great pleasure working along with all the passionate researchers at Institute of Neuroinformatics in a cosy and collaborative environment. Actually, INI became my second home for a substantial period of my life, and I am grateful to all the people who made the institute such a comfortable place to work at.

My friends — Peter Dziennik, Yulia Sandamirskaya, Alexey Illarionov, Hongjie Liu, Diethelm Raff, Valeriy Petrunin, Olga Krivets were encouraging me at different stages of my work and life.

And finally, I am thankful to my soulmate and girlfriend Zhenya, who supported and motivated me over the last year of this project. Zhenya, I would not be able to finish this work without your permanent care, love and patience.

CONTENTS

1 INTRODUCTION 3
1.1 Background on Spiking Sensors and Event-driven Hardware 3
1.2 Background on Local AGC 4
1.3 Background on Multi-microphone Platform for Ad-hoc Network . . 6
1.4 Thesis Outline 8

2 EVENT-DRIVEN HARDWARE SYSTEM FOR SENSOR FUSION 11
2.1 Introduction 11
2.2 Methods 12
2.2.1 Setup 12
2.2.2 Networks 13
2.2.3 Input Stimuli 14
2.3 n-Minitaur Architecture 17
2.4 Results 18
2.4.1 n-Minitaur Performance Measurements 18
2.4.2 Hardware Inputs 19
2.4.3 Single Sensor and Sensor Fusion Experiments 20
2.5 Discussion 21

3 AUTOMATIC GAIN CONTROL WITH SPIKING COCHLEA 25
3.1 Introduction 25
3.2 Methods 26
3.2.1 DASLP Silicon Cochlea 26
3.2.2 Gain Switching Transient Response 28
3.2.3 Sensor Features 31
3.2.4 Dataset Preparation 31
3.3 Event-driven AGC Algorithm and Hardware Implementation 32
3.3.1 Spike-driven Local AGC Algorithm 32
3.3.2 FPGA AGC Implementation 34
3.4 Results 36
3.4.1 Dependence of Analog Filter Output on Input Amplitude . . 36
3.4.2 AGC Steady-state Spike Response Measurements 44
3.4.3 Spike Frequency Responses for non-AGC and AGC Cases . . 46
3.4.4 AGC Transient Measurements to Speech 52
3.4.5 Input Features and Classifier 55
3.4.6 Voice Activity Detection Classification Task 55
3.5 Discussion 58
3.5.1 Improvements of Local AGC Mechanism 59

4 MULTI-MICROPHONE PLATFORM FOR AD-HOC NETWORK 61
4.1 Background 61
4.1.1 Challenges of Platform Development 62

4.2 WHISPER platform . . . 64
4.2.1 WHISPER Hardware . . . 64
4.2.2 WHISPER Prototype Hardware . . . 67
4.2.3 FPGA Logic and Software . . . 67
4.2.4 Communication and Network . . . 68
4.3 Platform Synchronization . . . 70
4.3.1 Background of Synchronization Algorithms . . . 70
4.3.2 Synchronization Algorithm of the WHISPER Prototype Hardware . . . 70
4.3.3 Synchronization for the Final Hardware . . . 72
4.4 Beamforming Experiments with WHISPER . . . 78
4.4.1 Experiments with Prototype 1 . . . 78
4.4.2 Speech Separation Experiments with final WHISPER . . . 82
4.5 Discussion . . . 83

5 DISCUSSION AND CONCLUSION 85
5.1 Summary . . . 85
5.2 Outlook . . . 86

A APPENDIX 89
A.1 Supplementary Material to Chapter 2 . . . 89
A.1.1 n-Minitaur Implementation Details . . . 89
A.2 Supplementary Material to Chapter 3 . . . 93
A.2.1 Input Signal Conditioning Circuit . . . 93
A.2.2 Calibration of Input from Sound Card . . . 93
A.2.3 DASLP FPGA Control Logic Structure . . . 93
A.2.4 Implementation of ADC Data Transmission over Asynchronous AER Bus . . . 94
A.2.5 Embedding the AGC Channel Gain Information into DASLP Events . . . 99
A.2.6 jAER Control Panels and Biasgen Settings . . . 100
A.2.7 Measurements of Channel Characteristic Frequencies of DASLP 102
A.2.8 Effect of Local Gain Control on Responses to a Two-Frequency Component Signal . . . 103
A.3 Supplementary Material to Chapter 4 . . . 104
A.3.1 WHISPER Platform Implementation Details . . . 104
A.3.2 Phase Shift and Jitter of WHISPER Prototype . . . 108
A.3.3 Alternative Data Transmission Module . . . 109

BIBLIOGRAPHY 115

1 INTRODUCTION

Neuromorphic technology is based on the operating principles of the brain. Neuromorphic spiking sensors produce asynchronous streams of data in response to changes in the scene while conventional sensors such as cameras produce uniformly sampled image frames. Sensory processing algorithms that use neuromorphic spiking sensors are under investigation and their usefulness in natural environments are less explored in contrast to the more mature signal processing algorithms for uniformly sampled input. Testbeds for testing the networks and algorithms of sensory processing systems and determining advantages of these systems over conventional time-stepped systems would be needed for faster evaluation of the robustness of these algorithms in the wild. This thesis presents two different testbeds, one for evaluating neuromorphic spiking sensors and processors; and a second testbed for multi-microphone network.

1.1 BACKGROUND ON SPIKING SENSORS AND EVENT-DRIVEN HARDWARE

Spiking sensor designs, in particular the DVS (Li, Brandli, *et al.*, 2015; Lichtsteiner *et al.*, 2008; Posch *et al.*, 2014) and DAS (Liu, Schaik, *et al.*, 2014; Yang, Chien, *et al.*, 2016) have improved tremendously in recent years. The evolution of the different designs capturing some functionality of the biological sensors can be found in (Liu, Delbruck, *et al.*, 2015). Hardware spiking vision systems that use a DVS (Li, Brandli, *et al.*, 2015; Lichtsteiner *et al.*, 2008), including the spiking convolutional neural network system called CAVIAR (Serrano-Gotarredona *et al.*, 2009), have demonstrated their low-latency responses for real-time vision tasks such as tracking. There are fewer example systems that use a spiking cochlea sensor such as the DAS and hardly any that use both the spiking vision and cochlea sensors (Chan, Jin, *et al.*, 2012). In the first part of the thesis work carried out in 2015-2016, deep networks were starting to be applied towards the output spiking sensors and there were hardly any hardware test beds at the time to prototype spiking Deep Neural Network (DNN) in combination with both sensor modalities.

One of the earliest successes in training a deep network, in this case, a Deep Belief Network (DBN) so that it can be converted to a Spiking Neural Network (SNN), was reported in (O'Connor *et al.*, 2013). Newer conversion methods were then developed for the networks that came later and used the Rectified Linear Unit (ReLU) transfer function (Diehl *et al.*, 2015). This work reported conversion methods for small convolutional neural networks and full-connected multi-layer networks on the digit recognition dataset called MNIST. (Neil *et al.*, 2016) extended the study to include not only vision datasets but also audio datasets such as TIDIGITs. These early

networks used high precision bit values for the weight and bias parameters but this high precision is not available on many embedded platforms. A few groups started to develop quantization methods on the network parameters during training while still maintaining equivalent accuracy to the high-precision network on a specific task became important. The accuracy of the converted spiking network on a lower-precision hardware platform showed that the accuracy is equivalent to the accuracy of the higher-precision time-stepped Artificial Neural Network (ANN) (Stromatias, Neil, Pfeiffer, *et al.*, 2015).

These converted spiking networks can be implemented on hardware spiking network platforms such as SpiNNaker developed at the University of Manchester. These networks performed well on a recognition task even with inputs coming from a spiking DVS. The SpiNNakeR platform was used to implement a deep network that was trained on a digit recognition task and uses as inputs, the spike outputs of a DVS sensor (Stromatias, Neil, Galluppi, *et al.*, 2015).

The SpiNNaker platform was built to support large scale spiking networks. To evaluate smaller networks using more available embedded platforms, the authors in (Neil *et al.*, 2014) implemented a real-time event-driven FPGA-based (Spartan 6) spiking hardware accelerator called Minitaur.

This accelerator could run a spiking deep network which achieves 19 million postsynaptic currents per second and supports up to 65 K neurons per board. One goal of the Minitaur hardware development was to implement a hardware system that can implement the sensory fusion network in (O'Connor *et al.*, 2013) and to test its performance including latency when interfaced to the two sensor modalities. The work to interface Minitaur to a spiking sensor before I took on this thesis work, was still under development.

In this part of the thesis, I developed an improved spiking accelerator that works with the spiking sensors and that could respond with low latencies. At that time, there was little reported on the use of a sensor fusion system for an audio-visual task. The results of this system development are described in Chapter 2 along with the improvements made on the Minitaur architecture. We also conducted experiments with this hardware system, also called n-Minitaur, to determine the classification accuracy of the network on a well known digit dataset called MNIST. The network received input spikes from the DVS which was presented with digits from this dataset and spikes from the cochlea which was presented with an audio tone unique for each digit. Experimental results were reported on the network performance when only one modality is presented or both modalities are presented.

1.2 BACKGROUND ON LOCAL AGC

From the n-Minitaur experiments, it was difficult to operate the system under a wide range of audio amplitudes typically encountered in the real world. Therefore, we developed an automated gain control mechanism based on only the output spikes of the cochlea. While a global Automatic Gain Control (AGC) circuit on the microphone outputs (e.g. used on the DAS board (Liu, Schaik, *et al.*, 2014)) helps

the system to operate in far microphone settings, using an external component has certain drawbacks.

First of all, the external AGC consumes additional power and requires relatively high supply voltage compared to recent low-power cochlea design DASLP (Yang, Chien, *et al.*, 2016), Acoustic Feature Extraction (AFE) frontend (Goux *et al.*, 2020) or Voice Activity Detection (VAD) chips (Oh *et al.*, 2019; Yang, Yeh, *et al.*, 2018). E.g., the MAX9814[1] chip, used in (Liu, Schaik, *et al.*, 2014) has $\sim$ 10 mW power consumption and requires at least 2.7 V power supply, while the AFE of the DASLP chip consumes only 55 μW at 0.5 V power supply and AFE of the VAD chip (Yang, Yeh, *et al.*, 2019) consumes under 1 μW at 0.6 V. So, for many applications involving low-power acoustic sensors, such as VAD, Keyword Spotting (KWS), voice control, anomaly detection and predictive maintenance (Coady *et al.*, 2019), it would be beneficial to have an on-chip low-power AGC.

Second, the global AGC estimates the power of the whole signal and amplifies or attenuates all frequency bands by the same amount. However, in many applications there could be significant amount of noise outside of the frequency band of interest. In this case, the global AGC would attenuate both, the noise and the useful signal, pushing the latter below a detection threshold.

Meanwhile, it has been known for a long time, that the outer hair cells in biological cochleas have local gain control mechanism (Ruggero, 1992) that allows animals to hear soft sounds in a presence of loud out-of-band noise. We suppose that the artificial spiking cochlea will benefit from a local AGC mechanism similar to one observed in biological cochleas. Various ASIC implementations of cochlea with local AGC models have been reported (Wen *et al.*, 2009) (Yang, Lyon, *et al.*, 2015) (Hamilton *et al.*, 2008) (Moeys *et al.*, 2015). Because of the complexity of local AGC models and transistor mismatch, it is not easy to build an ASIC cochlea circuit within a reasonable chip area and with good matching across multiple filter channels. Other hardware implementations include one that implemented the biomorphic Hopf model using discrete electronics (Vyver *et al.*, 2003). Another recent reported implementation was the FPGA implementation (Xu *et al.*, 2018) of the CAR-FAC model (Lyon, 2011) which incorporates recent findings on cochlear wave mechanics. The latter two systems do not have to deal with transistor mismatch but consume more power and are larger than the ASIC circuits.

In Chapter 3, we describe a new method for implementing the desired local AGC mechanism applied to one of the spiking cochlea ASICs. In this case, we used the more recent low-power binaural DAS called the DASLP (Yang, Chien, *et al.*, 2016) which has a very low power consumption of 55 μW from the analog filter bank core and good matching across the 64 filter channels. The low power consumption of this cochlea is partly due to the low power supply of 0.5 V for the analog filter bank circuits. Because of this low power supply and the transistor operation to implement the desired transfer function, the input signal dynamic range for the circuits to behave well is quite small. However, the DASLP chip has an integrated programmable attenuator and a Programmable Gain Amplifier (PGA) for each of its 64 channels. The combined gain of a channel ranges from 0 dB to about 40 dB. Also, the DASLP provides a digital interface for altering the channel gain while operating,

1 https://datasheets.maximintegrated.com/en/ds/MAX9814.pdf

which is fast enough for using in a feedback loop. Thus, updating a channel gain based on the current amplitude at the output of the filter would extend the dynamic range by 40 dB. For comparison, the dynamic range of the aforementioned external AGC chip MAX9814 is only 20 dB.

Because there is no direct access to the outputs of all the cochlea channels, the signal amplitude at the output of a filter can only be estimated by the spikes produced by the channel. We study how estimation through spike counting impacts the stability of the control loop. We utilise the FPGA of the DASLP board for implementation of our AGC algorithm, so no additional components are required and the latency of the gain control loop is kept low. In a further study, we investigate whether the spikes of the AGC-controlled cochlea can help improve the accuracy in an example audio task of classifying speech versus noise.

1.3 BACKGROUND ON MULTI-MICROPHONE PLATFORM FOR AD-HOC NETWORK

The final part of this thesis covers the development of a multi-microphone acoustic sensor platform called WHISPER, which belongs to conventional signal processing domain involving uniform audio sampling. This work was done within a multi-partner European project called COCOHA (COgnitive COntrol of a Hearing Aid). The main objective of this project was to study the possibility of using EEG signals from a hearing aid user in order to amplify the sound source of interest and suppress all the other sounds to some extent. Therefore, a device that can selectively amplify one of the sound sources from an auditory scene in real time was needed. This objective also imposes a low-latency requirement because it is important to switch between the sound sources promptly and deliver the selected sound with minimal delay in order to preserve an audio-visual speech synchrony.

One way of amplifying the sound source of interest is through beamforming methods that are used to locate a sound source (e.g. a talker) using a set of microphones (Van Veen *et al.*, 1997). These methods usually depend on a good characterization of the time difference of sound waves arriving at a set of microphones with a fixed geometrical arrangement. The signal-to-noise ratio (SNR) of a source can also be improved acoustically by placing the microphone closer to the source. However, a distributed ad-hoc network of microphones provides the most flexible solution as in distributed wireless acoustic sensor networks (WASN) (Bertrand *et al.*, 2015) because they impose less stringent restrictions on the microphone placement compared to fixed microphone arrays or placing a separate microphone next to each sound source.

Increasing SNR of the target source with a spatially distributed ad-hoc set of microphones is especially useful for hearing aid devices that have only two to four microphones that are spaced closely together; and for ambient intelligent space monitoring.

In general, a WASN that supports as many distributed microphones as possible within a space will allow more spatially coherent sources to be isolated with shorter impulse responses (Benesty *et al.*, 2007), and provide better rejection of diffuse

noise (Levin *et al.*, 2015). It is also more likely that at least one microphone will be close to the target or to a major noise source that needs to be factored out. Computation capabilities should be added at each node so that bandwidth usage is reduced and the platform can support distributed processing algorithms. Each node would then transmit a smaller number of processed signals instead of transmitting all its microphone signals simultaneously to the processing node. It also means that the network can be scaled up by adding more nodes without a noticeable impact on the overall computational load and bandwidth usage of the system (Bertrand *et al.*, 2015).

With local computing power, a node can perform within-node processing on the signals of their own microphones, and possibly together with signals received from neighboring nodes. A node that "sees" a source with the best SNR could "own" the source (Markovich-Golan *et al.*, 2012) therefore allowing the source to be cleaned before it is made available for other nodes. Within-node processing can also benefit from signals transmitted from those nodes that own the noise source. Other forms of within-node preprocessing include inter-stream correlation that achieves a degree of compression of the signals and furthermore, once a filter solution has been computed, each node then needs to transmit only one stream to others or to the hearing aid (Bertrand, 2011). Thus, the network can deliver to the user's ears a signal with a better SNR than any individual microphone.

With current technology, one can construct a portable platform with enough computational power for the aforementioned local processing by using devices such as Field-Programmable Gate Array (FPGA) and embedded computers. We envision the WHISPER platform to be a modular platform constituted by up to four identical WHISPER-M4 modules carrying a CPU, an FPGA, four microphones and a wireless connectivity module. Such a platform can then be used for prototyping real-time sound processing algorithms such as spatial sound filtering, noise cancellation and blind source separation. Some of these algorithms require synchronized microphone samples from the network, therefore one of the main challenges in building a useable WASN platform is the issue of sampling clock synchronization.

One viable solution that helps to avoid the synchronization issue is developed in (Maat *et al.*, 2014) for zebra finch songs recording from several individual birds at the same time. It involves using analog microphones with Frequency Modulation (FM) transmitters and a multi-channel Analog to Digital Converter (ADC) at the receiving site. However, having 16 analog FM microphones and 16 radio receivers attached to a multi-channel data acquisition system is more cumbersome than having 4 compact modules with 4 microphones each, that can form an ad-hoc Wi-Fi network together with a laptop. So, we opted for a completely digital solution relying on wireless synchronization.

Prior work in clock synchronization across multiple nodes of a distributed multi-microphone network includes the realigning of the clocks based on acoustic cues, wireless synchronization, GPS, or blind synchronization techniques (Girod *et al.*, 2006; Miyabe *et al.*, 2015; Schörkhuber *et al.*, 2014). One study tested the latter method in an off-line multi-talker speech recognition task using an ad-hoc network of smartphones and the cloud (Ochi *et al.*, 2016). The solutions in (*3D Audiosense* 2014; Girod *et al.*, 2006; Schörkhuber *et al.*, 2014) reported synchronization precision

in order of tens of microseconds but none of them addressed low-latency data transmission.

A big reason for developing the WHISPER platform was a necessity to have a versatile and flexible testbed for prototyping multi-channel audio processing algorithms and testing them in the wild. Although it is possible to simulate different acoustic environments in an anechoic chamber and perform recordings with a set of stationary microphones, such experiments are lengthy and costly. It is also difficult to incorporate all the required equipment into a low-latency signal processing pipeline with a feedback from the EEG based attention analysis. The secondary goal was to build a portable wireless device which can be used in conjunction with a traditional hearing aid to improve intelligibility of a speaker of interest and suppress all surrounding noises. Hence the name of the platform, WHISPER stands for "Wireless Hearing Improves Speech PERception".

In order to provide a test bed for developing and testing beamforming algorithms on an ad-hoc WASN, we built the first prototype of the WHISPER platform from the commercially available off-the-shelf (COTS) components only. The core of the WHISPER-M4 module is comprised of a low-cost Raspberry Pi 3B+[2] single board computer (SBC), which has wireless capabilities, connected with a low-cost FPGA board[3]. The main challenge was finding a synchronization algorithm that provides small enough sampling clock phase differences across multiple modules even in an ad-hoc arrangement. We developed a wireless synchronization method that utilizes a separate transceiver module[4] operating in industrial, scientific and medical (ISM) frequency range.

Our solution for the wireless synchronization is described in Chapter 4. We evaluate the accuracy of the sampling clock phase synchronization provided by the proposed solution across four WHISPER-M4 modules and study the effect of desynchronization on a quality of a beamforming algorithm on the ad-hoc array of microphones.

1.4 THESIS OUTLINE

Chapter 2 describes the FPGA-based hardware platform, also called n-Minitaur (Kiselev, Neil, *et al.*, 2016a), which we designed to process in real-time the spikes from the DVS and DAS. It also describes the experiments that were carried out in a sensory fusion task using the inputs from both modalities.

In this chapter, we show the following contributions:

- The first demonstration of a real-time hardware system consisting of an event-driven spiking network FPGA implementation receiving input spikes directly from both a DVS and a spiking Dynamic Audio Sensor cochlea.
- The first demonstration back in 2016 of using sensor fusion from spiking audio and video sensors to improve classification accuracy of spiking deep neural network in noisy conditions.

2 https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus
3 https://valentfx.com/logi-pi
4 https://ww1.microchip.com/downloads/en/DeviceDoc/75017B.pdf

Chapter 3 describes a system-level per-channel gain control loop mechanism which uses the cochlea spikes to control the gain of the filter channel. Contributions include:

- The first automated gain control mechanism using the spike outputs of the frequency-specific filter channels of a Dynamic Audio Sensor spiking cochlea.
- Demonstration of improvement of accuracy of event driven classification task over wide dynamic range when using local event based gain control with frequency-specific time constants.

Chapter 4 describes the construction of a multi-microphone platform suitable of a Wireless Acoustic Sensor Network (WASN) and the synchronization protocol needed to guarantee low phase difference for the sampling clocks on all distributed modules. The contributions include:

- One of the first real-time multi-microphone hardware platform for a wireless acoustic sensor network that synchronizes the audio sampling clock on multiple spatially distributed modules through a wireless technique.
- The availability of this platform for evaluating multi-channel speech separation algorithms in the wild and results that show that this benchmarking is possible in natural spaces.

2 EVENT-DRIVEN HARDWARE SYSTEM FOR SENSOR FUSION

This chapter presents a real-time multi-modal SNNsystem implemented on an FPGA platform. The hardware DNN system, called n-Minitaur, demonstrates a 4-fold improvement in computational speed over the previous DNN FPGA system — Minitaur (Neil *et al.*, 2014). In addition, the system is interfaced directly to two different event-based sensor modalities: a Dynamic Vision Sensor (DVS) and a Dynamic Audio Sensor (DAS). The DNN for this bimodal hardware system is trained on the MNIST handwritten digit dataset, which was augmented by adding a unique audio tone for each class. When tested on the spikes produced by each sensor alone, the classification accuracy is around 70% for DVS spikes generated in response to displayed MNIST images, and 60% for DAS spikes generated in response to a mixture of the tones assigned to the digits and noise. The accuracy increases to 98% when spikes from both modalities are provided simultaneously. In addition, the system shows a fast latency response of only 5 ms.

A large part of this chapter comes from a paper titled "Event-Driven Deep Neural Network Hardware System for Sensor Fusion") presented at the IEEE ISCAS 2016 (Kiselev, Neil, *et al.*, 2016a) (Kiselev, Neil, *et al.*, 2016b). The author has only used the text related to the work contributed by the author in the papers.

2.1 INTRODUCTION

Hardware systems implementing spiking Deep Networks such as Convolutional Neural Network (CNN), Restricted Boltzmann Machines (RBM), and DBN have been demonstrated on hardware platforms including FPGAs (Neil *et al.*, 2014; Zamarreno-Ramos *et al.*, 2013), SpiNNaker (Furber *et al.*, 2014), and TrueNorth (Tsai *et al.*, 2016). Some of these systems have been tested on recordings from event-driven DVS (Lichtsteiner *et al.*, 2008) to demonstrate the advantages of event-driven computing. Hardware spiking vision systems interfaced directly to a DVS have been demonstrated for real-time applications. Although there are a few systems that are interfaced to a spiking cochlea sensor such as the Dynamic Audio Sensor (DAS) (Liu, Schaik, *et al.*, 2014; Yang, Chien, *et al.*, 2016) and to both visual and auditory modalities (Chan, Jin, *et al.*, 2012; O'Connor *et al.*, 2013), no system at the time had used a spiking DNN hardware in combination with both modalities.

This paper describes a real-time multi-modal DNN hardware system interfaced to two event-based sensors of different modalities. A spiking DNN was previously implemented on Minitaur, an event-driven FPGA-based (Spartan 6) spiking neural accelerator system. With this system, one can implement a spiking deep network which achieves 19 million postsynaptic currents per second (Neil *et al.*, 2014) and supports up to 65 K neurons per board. We describe the improvements made on the Minitaur architecture so that the new system called n-Minitaur is able to

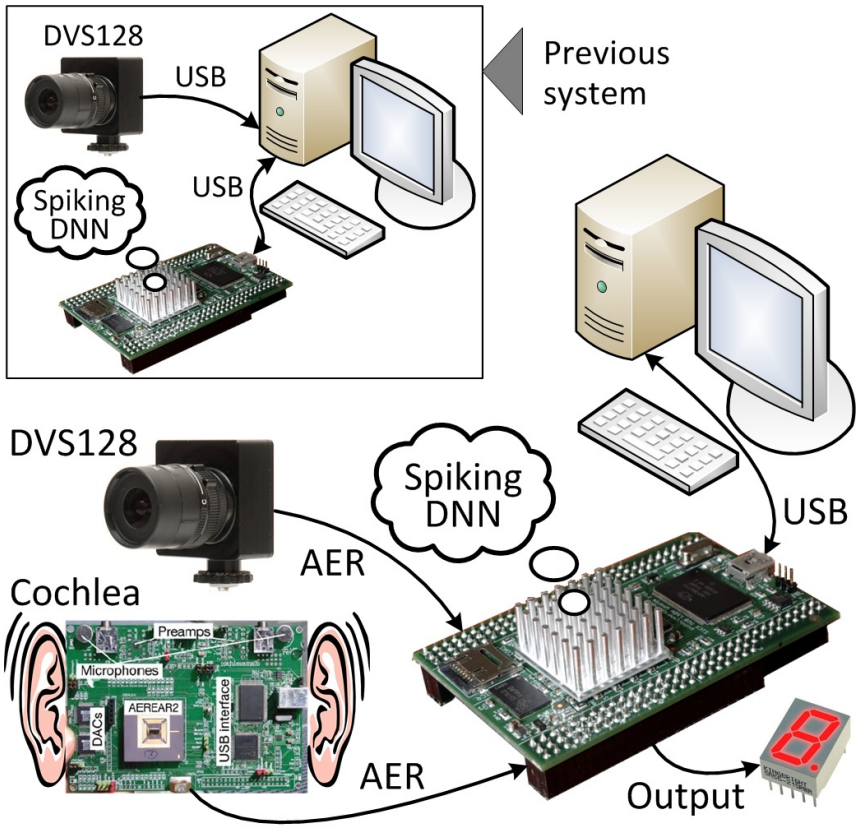

Figure 2.1: n-Minitaur system interfaced directly to both DVS and DAS sensors.

receive spikes in real-time from up to three event-based sensors. We conducted experiments using spiking DNNs implemented on this platform and demonstrated the classification accuracy on the MNIST dataset based on 1) the inputs from the two modalities separately and 2) the fusion of inputs from both modalities.

The setup is described in Section 2.2 followed by descriptions of two experiments using this platform for visual and visual-auditory tasks in Section 2.4 and a discussion in Section 2.5.

2.2 METHODS

In this section we describe the system setup consisting of an auxiliary board that holds an FPGA and interfaces to up to three different event-based sensors, two spiking network architectures, and input stimuli used in this work.

2.2.1 Setup

The system consists of a Spartan-6 FPGA board (ZTEX USB 1.15d[1]) and an auxiliary board with three Address Event Representation (AER) ports for direct interfacing to three spike-based sensors (Fig. 2.2). In this work, a DVS (Lichtsteiner *et al.*,

1 https://www.ztex.de/usb-fpga-1/usb-fpga-1.15.e.html

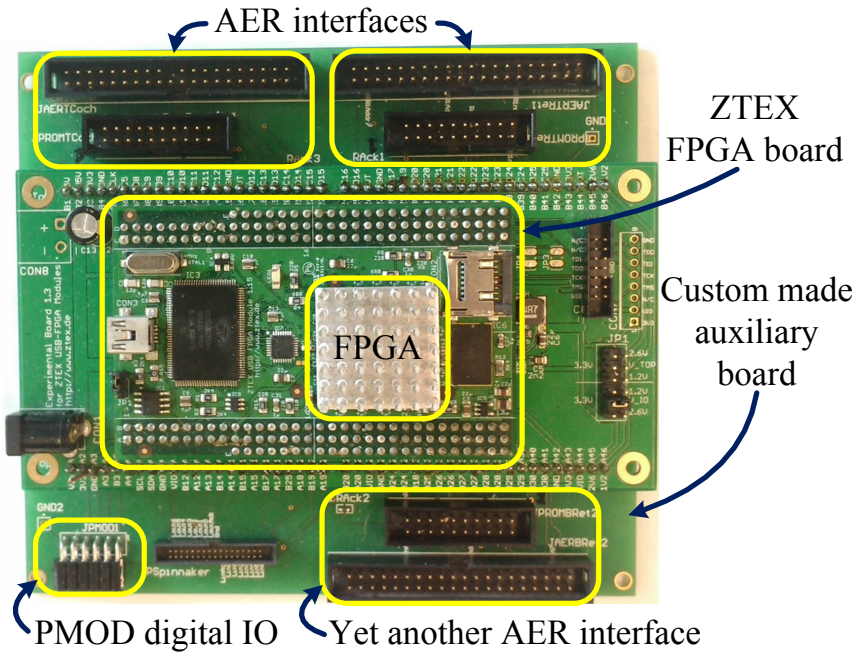

Figure 2.2: n-Minitaur interface board. Design from S-C. Liu and T. Delbruck.

2008) and a DAS (Liu, Schaik, *et al.*, 2014) are connected to two of these ports. The sensors communicate with n-Minitaur using an asynchronous AER protocol. Since one FPGA board handles the events from all the sensors in the order they arrive, timestamping of the events is not required at the sensors, and hence, no time synchronization is needed across the sensors in this setup.

The DVS128 retina, with a resolution of 128 $\times$ 128 pixels, produces events (spike addresses) only when a pixel senses local brightness changes. The pixels output ON and OFF events which encode both positive and negative changes in log-intensities respectively. The DAS board used in this work (DAS Ams1b) holds two microphones, a custom AEREAR2 binaural cochlea chip, and digital chips to handle the communication between the cochlea and the Personal Computer (PC). The board also has an external AER interface, which was adapted to the purpose of this work by modifying the AER acknowledge circuit, so that acknowledging an AER event generated by DAS would be possible from an external device. Each cochlea on the binaural AEREAR2 chip is modeled by a 64-stage cascaded second-order filter bank followed by a half-wave rectifier which models the inner hair cell and an integrate-fire neuron model which models the spiral ganglion cells.

2.2.2 Networks

Two DNNs are trained on the MNIST dataset following the training algorithm described in (O'Connor *et al.*, 2013). The first visual network (DNN1) of size 784-500-500-10 (shown in Fig. 2.3a) is trained on the 60,000 digit training set from the MNIST database. The second multi-modal network (DNN2) has additional 100 audio input neurons connected to the associative layer (Fig. 2.3b). These neurons are driven by the spikes of the DAS. The 64 cochlea channels of the DAS are mapped

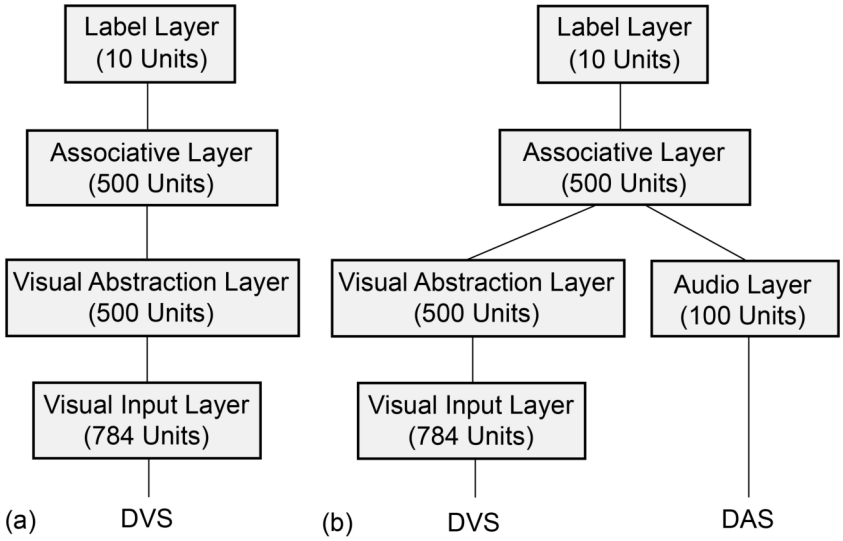

Figure 2.3: Network architecture of (a) the visual DNN1 and (b) the multimodal DNN2.

to the 100 neurons in the audio layer following the mapping scheme in (O'Connor *et al.*, 2013), which is described in the next subsection. Both networks are trained as DBNs, and then converted to feed-forward spiking neural networks (SNNs) for the classification task according to the algorithm proposed in (O'Connor *et al.*, 2013). The DNN1 network has 1794 Leaky Integrate-and-Fire (LIF) neurons with a total of 647,000 synapses. The DNN2 consists of 1894 neurons with 697,000 synapses. The LIF neuron model in the spiking DNN has a membrane potential decay time of 17.2 ms and a refractory period of 0.8 ms, as it was found to give the best performance for a practical range of the input event rates.

2.2.3 Input Stimuli

We describe next the two sets of test stimuli used in this work. The first set consists of artificial spike trains generated from the visual MNIST test dataset of 10,000 digits. In order to convert the static digit images to events, first, a number of the event addresses is generated with a probability proportional to the intensity of the pixel in the image of the digit. Then, for each event address a timestamp is generated uniformly within a defined time interval. The resulting events are sorted in a time increasing order and streamed to the hardware DNN over USB interface. A total of 1000 spikes are produced within 10 ms window for each digit. Thus, the average simulated spike rate is 100 kEvents/s

The second set consists of spikes streamed directly from the DVS and DAS sensors without a PC in the loop. The goal of this setup is to have a complete spiking DNN hardware system which performs the classification in real-time from the sensor spikes streamed directly to n-Minitaur. The setup is shown in Fig. 2.4.

For the visual spikes, the DVS128 is placed in front of an LCD display with an LED backlight. The LED backlight is preferable for the DVS compared to the fluorescent lamp backlight, because high frequency flicker of the lamp can create

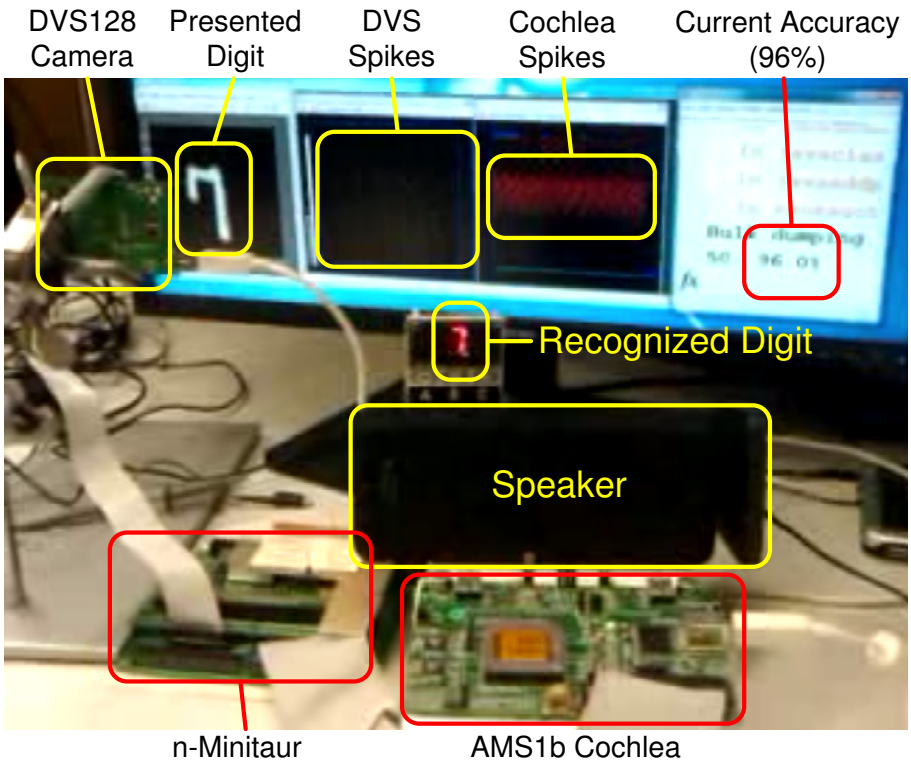

Figure 2.4: Setup of sensor fusion experiment.

additional events. The digits are presented on the screen at a frame rate of 30 Hz. Each digit is presented 8 times with each presentation alternating with a black screen.

For the audio spikes, a specific audio frequency is assigned to each digit similarly to (Neil *et al.*, 2014), but with slightly different frequency values (Table 2.1) for each digit for simpler on-FPGA computation – the frequencies are distributed in such a way that the corresponding periods are spaces linearly. Similar to the method in (O'Connor *et al.*, 2013), we preprocessed the DAS spikes on an FPGA with an event-based interspike interval (ISI) histogramming method where we built 100 ISI bins that were distributed linearly between 1.078 and 3.125 ms (320 – 928 Hz). Every input neuron in the Audio Layer (Fig. 2.3b) is assigned to one of the histogram bins. The corresponding neuron in this layer is stimulated when its assigned ISI bin receives an ISI value associated with a spike from any of the 4 output neurons within any DAS filter channel. An example of the spike sequence generated for an input stimulus for digit '3' is given in Fig. 2.5.

Both the audio tone and the image of a digit are presented simultaneously in the sensory fusion experiments, and the output spikes from both sensors are recorded at the same time via USB. The classification label is determined by the output neuron with the most spikes for each combination of tone and video digit.

Freq. (Hz)	352	371	385	414	444	470	552	588	670	720
Digit	0	1	2	3	4	5	6	7	8	9

Table 2.1: Tone-digit pairs in a multi-sensory fusion task.

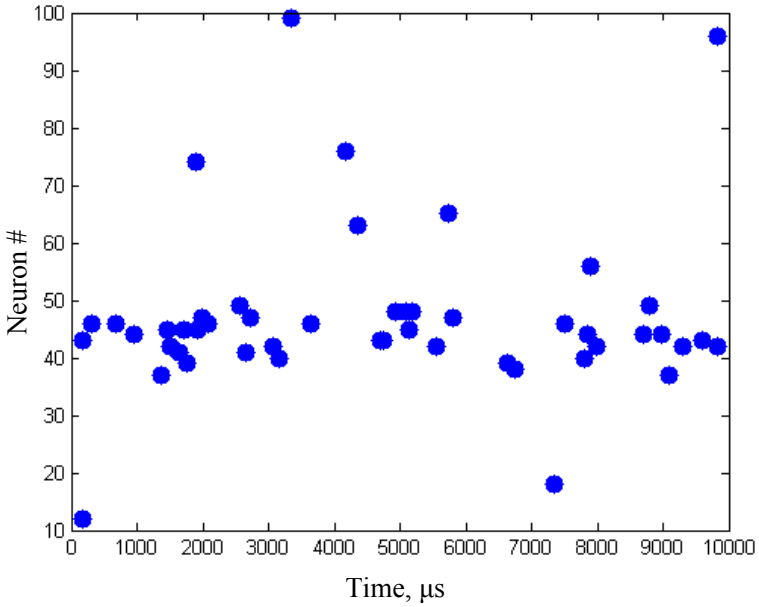

Figure 2.5: Audio stimulus events from the 100 neurons in the Audio Layer. Spikes are in response to the tone assigned to digit '3'.

In the real-time experiments described in Section 2.4, the hardware sensor spike rate is about 50 kEvents/s for the DVS and about 5 kEvents/s for the DAS. This spike rate exceeds the limit for real-time processing on n-Minitaur, which is 8.6 kEvents/s for the DNN2 architecture (see the last row of the Table 2.2). In order to achieve a real-time classification the spike rate is reduced by sending just 1 out of N input spikes to the DNN and dropping the rest N-1 spikes, where N ranges from 1 to 16. Both video and audio inputs are decimated using $N = 10$ and $N = 4$ respectively, to achieve an acceptable average event rate of approximately 6.5 kEvents/s. The decimated of the audio spikes is performed after mapping of the cochlea spikes to the ISI bins, so computing of the interspike intervals is not affected.

Because the accuracy of the DNN2 from just the DAS spikes is already close to 100% accuracy (see Table 2.4 5th row), an interfering frequency is added to the audio stimuli in order to reduce the classification accuracy of the network. First, the frequency for each digit d is selected randomly from a normal distribution with a mean at the central frequency $F_c(d)$ and a standard deviation of 1.7. Then, a second interfering frequency is chosen from a uniform distribution ranging from 300 to 800 Hz except for a region of $F_c(d) \pm 50Hz$. The two frequencies when combined together produce an audio signal with a signal-to-noise ratio of 6 dB. This set of audio stimuli corresponding to each digit produces significant error in the classification (see Table 2.4 row no. 7 in Section 2.4).

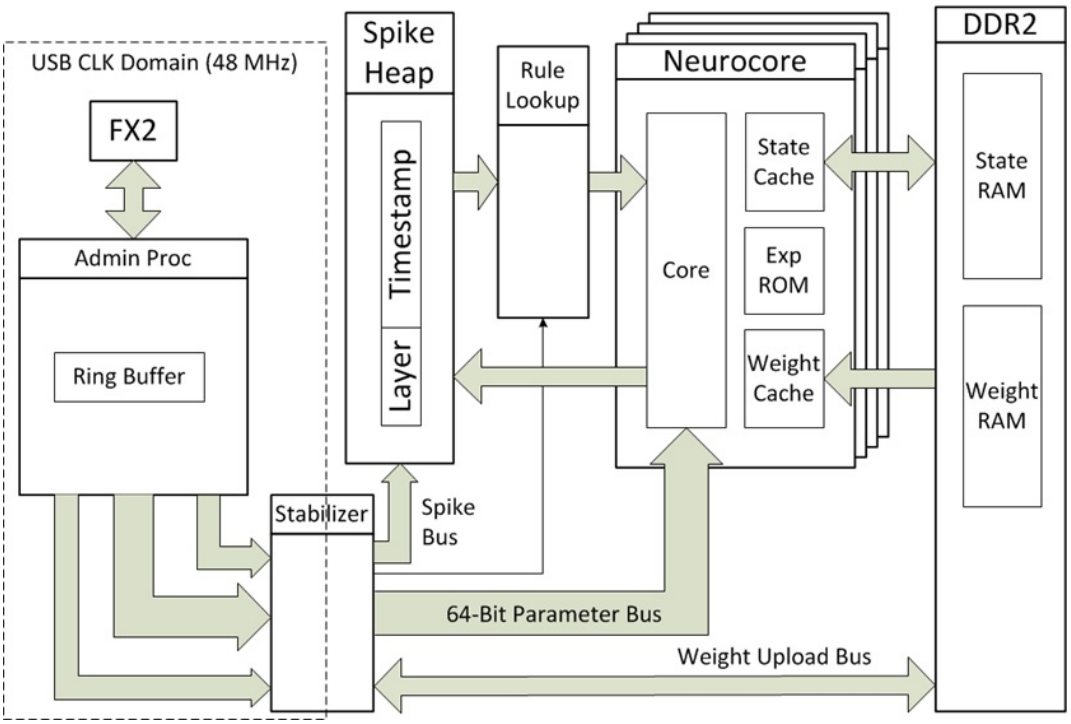

Figure 2.6: Original Minitaur.

2.3 N-MINITAUR ARCHITECTURE

Minitaur (Neil *et al.*, 2014) is implemented on a low-cost Xilinx Spartan-6 platform. The full implementation is done on a ZTEX USB 1.15d board, which holds 128 MB of DDR2 RAM, a microSD card slot for storage, 128-kB flash memory for a bootloader, and an FX2 chip for USB interfacing.

We improved the Minitaur architecture in Fig. 2.6 for lower system latency, higher throughput and the necessity for handling AER events from up to three event-based sensors. In order to process events arriving from multiple sensors, we modify the interface block in Minitaur which previously could only receive spikes from the DVS through a PC over USB2 bus (See the inset in Fig. 2.1.) The new architecture (n-Minitaur) is shown in Fig. 2.7, and its specifications are given in Table 2.2 in comparison with the original Minitaur.

The most significant improvements are described next. First, the USB interface and Control block (called "Admin Proc" in the original Minitaur design, Fig. 2.6) is redesigned completely. A pair of Xilinx Dual-Clock FIFOs is used to separate the USB and the accelerator clock domains in order to reduce amount of the USB-clock driven logic to a minimum. The Finite State Machine (FSM) controlling transmission of events between n-Minitaur and a PC in both directions is optimised, leading to a 4-fold increase of the USB data transfer bandwidth up to 30 MBytes/s. The block diagram (Fig. A.1), the FSM state transition diagram (Fig. A.2) and an example of the state transitions and control signals (Fig. A.3) are given in the Appendix section A.1.1.

Second, the state machines for the Spike Heap and Neurocore blocks and the weight caching strategy are redesigned for a 2.6-fold improvement of performance

at the same frequencies (Table 2.3, 2nd row). The DDR2 RAM interface frequency is increased by a factor of 2, up to 264 MHz. Weight storage in the RAM is optimized, allowing uploading of a twice bigger network. Due to a linear arrangement of the neuron weights in the memory it was possible to replace the weight caching algorithm by weight prefetching using a burst mode of the DDR2 memory.

Third, by introducing the registered Parameter Bus, the design complexity and congestion level are reduced, and the core frequency is increased from 75 MHz to 105 MHz, leading to a 4-fold overall increase of performance.

In addition, the input multiplexer, the ISI histogram computation and the event decimation blocks are implemented, allowing real-time sensor fusion of different event-based sensors. The classification result is displayed on a 7-segment LED indicator attached to the PMOD connector of the interface board.

The n-Minitaur neural accelerator operates as follows. The input spikes are timestamped at the input multiplexer and placed into a heap, which is sorted by the timestamp. The spike with the smallest timestamp is taken from the top of the heap and is sent to the connection rule lookup block. Meanwhile the heap is updated such that the spike with the next smallest timestamp appears on the top. The update time is proportional to the height of the heap. The spike carries information about the neuron that generated this spike. The lookup table is searched for the connection rule that has this neuron at the source side. The output side of the connection rule represents a range of neurons in the subsequent layer that are connected to the source neuron, so the membrane potential of these neurons has to be updated when the source neuron emits a spike. When the connection rule is found, the neurons of the range defined by this rule are distributed among 32 neurocores. Each neurocore computes a membrane potential decay based on the time passed since the last membrane potential update of the neuron and adds a value defined by the weight of its connection with the source neuron. Then the membrane potential is compared to a threshold and if the threshold is exceeded, the membrane potential is set to 0 and the spike is generated by the current neuron. The new spike is timestamped and placed into the spike heap. The time of the membrane potential update is stored in the State RAM along with the new value of the membrane potential. When the update process is finished for all the neurons of the aforementioned range the next spike is picked from the top of the heap and the process repeats.

2.4 RESULTS

This section describes the specifications of the improved n-Minitaur implementation and the sensor fusion classification experiments using n-Minitaur.

2.4.1 n-Minitaur Performance Measurements

The architecture improvements of n-Minitaur give 4-fold increase of the compute speed compared to a previous version, while keeping the same classification accuracy. Table 2.3 (2nd row) shows the architectural performance improvement of

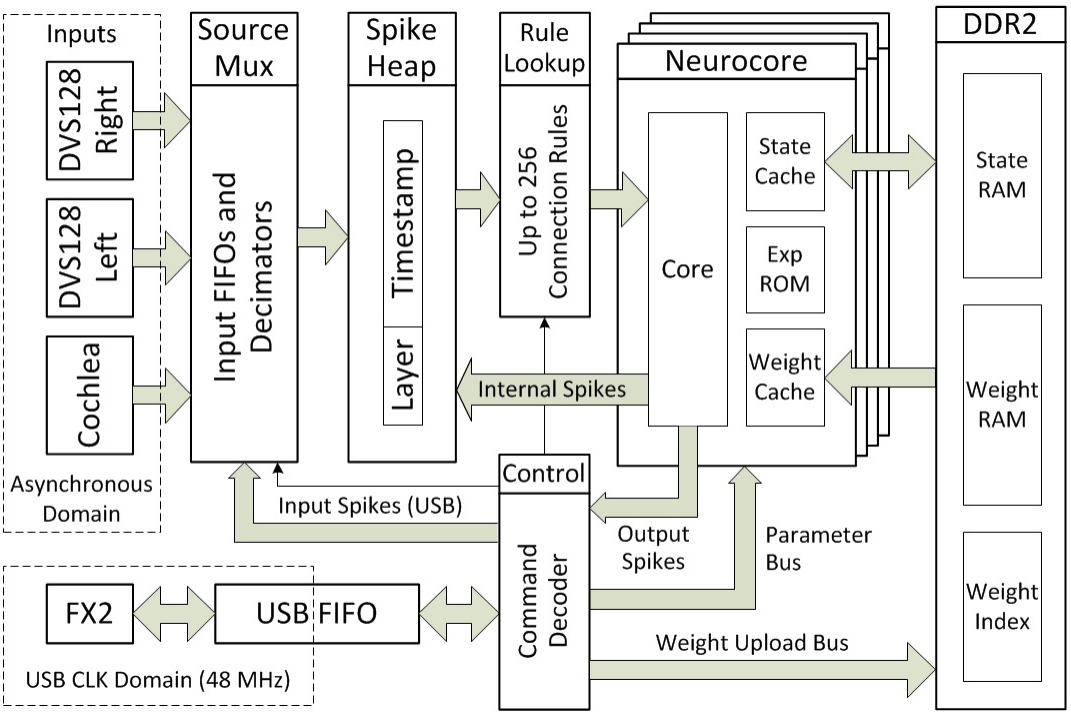

Figure 2.7: New architecture of Minitaur (n-Minitaur).

n-Minitaur at the same operating frequency as Minitaur. The operation time on the test set of the MNIST dataset is redused by a factor of 2.6. The new architecture also allows to increase the operating frequencies of the Neurocores and the external RAM to 105 MHz and 264 MHz, respectively, leading to a further 1.5-fold performance improvement. Further increase of the core frequency up to 132 MHz does not give much gain of performance due to a bottleneck in the external memory interface, and in addition, it leads to a significant growth of the FPGA design place-and-route time. Therefore, we decided to stick to 105/264 MHz Core/RAM frequencies for all the following experiments.

2.4.2 Hardware Inputs

Although Minitaur is capable of processing events streamed separately from event-driven visual and audio sensors using the USB interface, it was impossible to combine events from different sensors in real-time because the USB stack introduced unpredictable delays of hundreds of μs while events are captured with 1 μs precision. Moreover, the spike data have to be packetized for USB transaction, so enough events have to be collected to initialize the data transfer. This introduces additional delays. On the other hand, the hardware AER interface allows to stream events from different AER sources into a processing module in real-time, immediately at the time the events are generated.

To ensure that input spikes arrive to the network in the correct spatial arrangement, the spikes from the first layer of the network are recorded and displayed (Fig. 2.8). The images show that the spikes are transmitted properly to the FPGA system.

Parameter	Original (Minitaur)	Modified (n-Minitaur)
Neural accelerator parameters		
Number of neurocores	32	32
Max number of neurons	65536	65536
Max number of synapses	16.78 M	33.5 M
Max number of connection rules	128	256
Design statistics		
Nets	173500	101255
Slice registers	17425	11669
Slice LUTs	23778	15137
Occupied slices	8845	5201
DSP blocks per neurocore	2	1
Timing specifications		
Operating frequencies (Core/RAM)	75/132 MHz	132/264 MHz
USB download bandwidth	7.4 MB/s	30.3 MB/s
100 events download time	116 us	26 us
Min input to output (I-O) latency	293 us	238 us
I-O latency on MNIST DNN (mode)	9.2 ms	2 ms
Serial processing speed (updates/s)	0.59 Mupd/s	1.67 Mupd/s
Peak processing speed (updates/s)	18.9 Mupd/s	53.5 Mupd/s
Input spike processing rate on MNIST DNN	1.95 kSpikes/s	8.6 kSpikes/s

Table 2.2: Specifications of Minitaur and n-Minitaur.

FPGA logic version	Clock (MHz)		Operation time (s)	Time per digit (s)	Accuracy (%)
	Core	RAM			
Original	75	132	5390	0.539	92.0
Modified (original frequency)	75	132	**2084**	0.208	92.06
Modified (stable result)	105	264	**1372**	0.137	92.08
Modified (max frequency)	132	264	1359	0.136	92.04

Table 2.3: Performance of n-Minitaur with DNN1 on the MNIST dataset (10000 digits).

2.4.3 Single Sensor and Sensor Fusion Experiments

The hardware implementation of the DNN1 on n-Minitaur is first tested on artificially generated spike-trains from the MNIST database and led to an accuracy of 94.1% (Table 2.4, Expt No. 1) comparable with a software simulation of the same

#	Experiment description	% of correctly labelled digits
1	DNN1, visual only, artificial MNIST spikes	94.1
2	DNN1, DVS spikes	89.9
3	DNN2, visual only, artificial MNIST spikes	81.5
4	DNN2, visual only, DVS spikes	70.0
5	DNN2, audio only, artificial spikes	99.9
6	DNN2, audio only, cochlea spikes	99.0
7	DNN2, audio only, cochlea + noise	60.0
8	DNN2, DVS + (cochlea + noise)	98.0

Table 2.4: Table of accuracies on the MNIST dataset and audio tones.

network (Neil *et al.*, 2014). This accuracy decreased by 4.6% (Table 2.4, Expt No. 2), when tested with DVS spikes streamed directly into n-Minitaur through the AER interface. This decrease can be explained by a difference in the spike statistics between artificially generated spikes and real DVS spikes. The multi-modal network DNN2 is first tested on visual spikes using either the artificial visual spikes or the DVS spikes (Table 2.4, Expts No. 3, No. 4). The accuracy of the DNN2 is much lower than that of the DNN1 in this case. But the accuracy of the DNN2 using the audio input alone (Table 2.4, Expts No. 5, No. 6) is around 99.9%, suggesting that the DNN2 network learned to rely more on the audio input. Our hypothesis is that it was partially due to over-simplified audio stimuli (pure tones) used to represent the digits.

By adding an audio noise, the accuracy of the network with audio-only input decreased to 60% (Table 2.4, Expt No. 7). However, by fusing spikes from the visual and the audio sensors when the audio signal is mixed with the noise, classification accuracy was restored to 98% (Table 2.4, Expt no. 8), demonstrating that sensory fusion can help improve performance of the system which performs poorly with only a single modality.

2.5 DISCUSSION

This work aims to create a complete spiking hardware DNN system capable of processing spikes from event-based audio and visual sensors. This embedded hardware system can be interfaced to a maximum of 3 AER sensors without a PC in the data path. However the PC is still required for uploading the network structure and coefficients, and for configuring the sensors. The need for a PC can be eliminated by using modern System on a Chip (SoC) FPGAs, such as Xilinx Zynq Ultrascale+, which has on-chip CPU cores.

The performance of the new DNN FPGA architecture (n-Minitaur) has been improved by a factor of 4 along with a reduction in resource utilization of 35%. The performance of the system in classifying the visual MNIST digits is of similar accuracy to that of Minitaur (Neil *et al.*, 2014). The hardware network implemen-

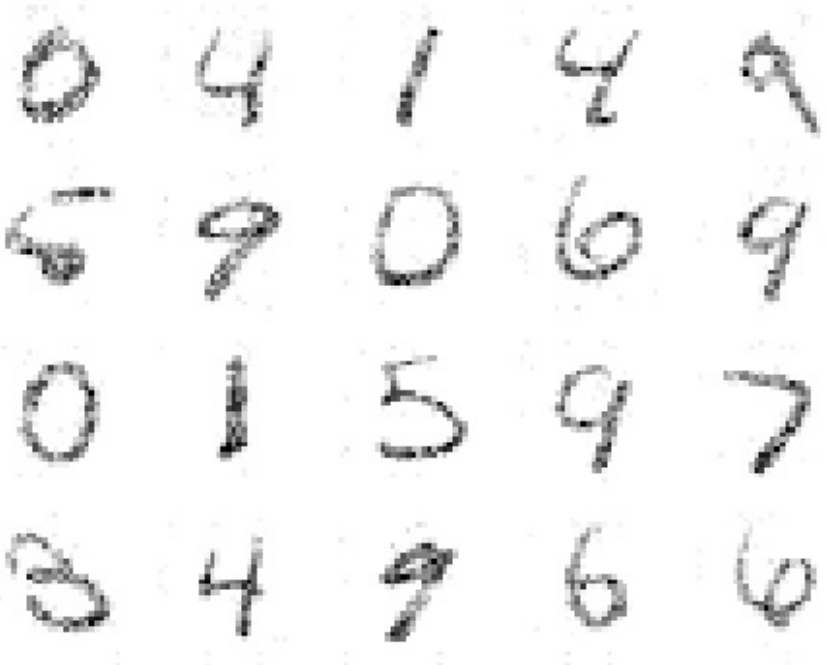

Figure 2.8: Spike histograms recorded from the first layer of the network connected to the DVS via the hardware AER interface. Digits are presented to DVS as described in Section 2.2.3. 1000 events are recorded for each digit.

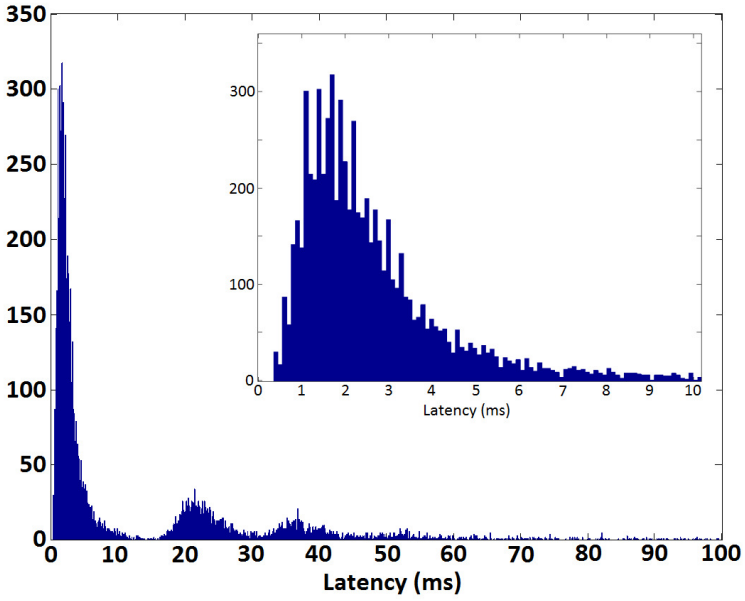

Figure 2.9: Histogram of latencies from the first input spike to the first output spike based on the 10000 test digits of the MNIST dataset.

tation shows a classification accuracy increase when using input events from two event-based sensors of different modalities.

The measured latency between the 1st input spike and the 1st output spike is only about 5 ms (Fig. 2.9), when tested on the MNIST digit spikes, showing a real-time operation of the system. The first and the biggest mode of the latency distribution is just around 2 ms. The 2nd and the 3rd modes, at $\sim$ 22 and $\sim$ 37 ms respectively, are likely due to the fact that the input spikes for the latency measurement are sent over USB. Although the latency measurement is performed on the FPGA, interruption of the data uploading thread by the Windows task scheduler between two USB transactions can cause the increase of the latency. If input spikes in the first data packet has not elicited any output spikes, the accelerator has to wait for the second packet, however, the latency measurement has already started at the beginning of the first packet. Thus, the data transmission delay can be included into the accelerator latency measurement leading to a heavy tail in the latency distribution.

We developed a hardware event-driven system that achieves real-time classification at event rates up-to 8.6 kEvents/s on the network with $\sim 700,000$ synapses. The amount of computations in the developed accelerator is by design proportional to the number of the network coefficients (synapses) and to the input spike rate. For many low power applications it is difficult to fit a high performance accelerator within a given power budget, and reducing the networks size usually leads to the classification accuracy loss, so reducing a spike rate might be a viable option for achieving a real-time performance. We demonstrated that high classification accuracy can be achieved even when the spike stream from a sensor is decimated by a factor of 10. However, the sensors are still generating lots of events within this approach. Reducing the number of events transmitted from a sensor to a processing device might further improve the power consumption of the system. In the following chapter we study a possibility of reducing the event rate produced by the DAS Low Power (DASLP) audio sensor by implementing a spike-based local gain control.

3 AUTOMATIC GAIN CONTROL WITH SPIKING COCHLEA

This chapter presents a real-time automatic gain control (AGC) system which uses the spiking output of a Dynamic Audio Sensor (DAS) to control gain settings of the bandpass filter bank. A part of the chapter comes from a paper titled "Event-driven Local Gain Control on a Spiking Cochlea Sensor" that will be presented at the 2021 IEEE ISCAS (Kiselev and Liu, 2021).

3.1 INTRODUCTION

Application-Specific Integrated Circuit (ASIC) cochlea designs implement circuits that model part of the functionality of biological cochleas Lyon *et al.*, 2010. More recent designs include circuits that generate asynchronous spiking outputs (Abdalla *et al.*, 2005; Chan, Liu, *et al.*, 2007; Liu, Schaik, *et al.*, 2014; Wen *et al.*, 2009; Yang, Chien, *et al.*, 2016). Few designs have included the local automatic gain control (AGC) properties of the outer hair cells in the biological cochlea (Hamilton *et al.*, 2008; Katsiamis *et al.*, 2009; Wen *et al.*, 2009). Local AGC allows the cochlea to be operated in natural uncontrolled environments where sound amplitudes can vary over a large amplitude range (Ruggero, 1992). Because of the complexity of local AGC models and transistor mismatch, it is not easy to build an ASIC cochlea circuit within a reasonable chip area and with good matching across multiple filter channels. A design with over hundred filter stages can suffer from mismatch that makes it less usable (Wen *et al.*, 2009) or the resulting pixel size of each channel prohibits the implementation of large number of channels on one chip (Katsiamis *et al.*, 2009).

Other hardware cochlea systems with AGC include the biomorphic Hopf cochlea system implemented using discrete electronics (Vyver *et al.*, 2003); and the FPGA implementation (Xu *et al.*, 2018) of the CAR-FAC AGC model (Lyon, 2011). These systems, while not having to deal with transistor mismatch, consume more power and are larger than the ASIC models.

Hardware spiking cochleas, e.g. the Dynamic Audio Sensor (Liu, Schaik, *et al.*, 2014), and software cochlea models are being tested as front-ends for audio tasks. These tasks include the using of the spike timing for azimuthal source localization (Anumula, Ceolini, *et al.*, 2018; Finger *et al.*, 2011; van Schaik *et al.*, 2009), speech recognition (Gao, Braun, *et al.*, 2019; Wu *et al.*, 2018), speaker verification, multi-modal recognition (Kiselev, Neil, *et al.*, 2016a) and keyword spotting (Ceolini, Anumula, *et al.*, 2019).

Spike features such as constant time bin and constant spike count features (Acharya *et al.*, 2018; Anumula, Neil, Delbruck, *et al.*, 2018) are usually presented as inputs to a machine learning classifier such as the SVM classifier (Li, Delbrück, *et al.*, 2012) or to deep neural networks (DNNs) such as recurrent neural networks.

These features have been applied quite successfully on the spiking cochlea outputs for real-time operation (Anumula, Neil, Li, *et al.*, 2017; Gao, Braun, *et al.*, 2019). The spikes can also directly drive a converted spiking deep network (Kiselev, Neil, *et al.*, 2016a).

In these different application domains where the input amplitude range can vary over 60 dB, the system will benefit from a local AGC mechanism in addition to a global AGC circuit used on the microphone outputs (e.g. on the DAS board (Liu, Schaik, *et al.*, 2014)). This global AGC helps the system to operate in far microphone settings, however it can cause unwanted attenuation of a useful signal in presence of noise outside of the band of interest.

Because of the difficulty of including the local gain control circuits on-chip, this chapter presents a system-level AGC mechanism that uses instead the spiking activity of the individual filter channels on a DASLP (Yang, Chien, *et al.*, 2016) to adapt the local gain of the filters. Although there are various methods that could be used to implement a negative feedback control algorithm, we focus on the possibility of implementing a local AGC mechanism that does not need floating or even fixed point arithmetic. The AGC mechanism proposed in this chapter needs only counters and adders thereby making it possible for future implementation in an ASIC. In this chapter, we describe the AGC experiments using the DASLP cochlea and a classification experiment of speech versus noise to evaluate whether adding a local gain control mechanism to a spiking cochlea cochlea will help to increase the classification accuracy of a classifier trained on the spike features compared to non-AGC case.

We describe first in Section 3.2, details of the DASLP cochlea architecture and bias control settings, the sensor features and the dataset preparation for the audio classification task. Section 3.3 presents the AGC algorithm and the corresponding implementation on the FPGA hardware platform that interfaces to the cochlea. Measurements of the analog and spiking responses of a filter channel with AGC are presented in Section 3.4 along with a classification experiment of speech vs noise for a large range of sound amplitudes.

3.2 METHODS

3.2.1 DASLP Silicon Cochlea

The DASLP spiking cochlea used in this work is the latest low-power (LP) ASIC binaural design with 64 frequency channels per side and asynchronous spiking outputs (Yang, Chien, *et al.*, 2016). This cochlea design uses a parallel bank of 64 filters ranging from best characteristic frequencies of 20 Hz to 20 kHz. The best frequency of each filter is generated by the 64 geometrically-scaled current block in Fig. 3.1. The fourth-order bandpass filter (BPF) design in each channel consists of two cascaded power-efficient second-order source-follower-based BPFs, followed by an asynchronous delta modulator (ADM) with on-chip asynchronous arbitration circuits for transmitting events off chip.

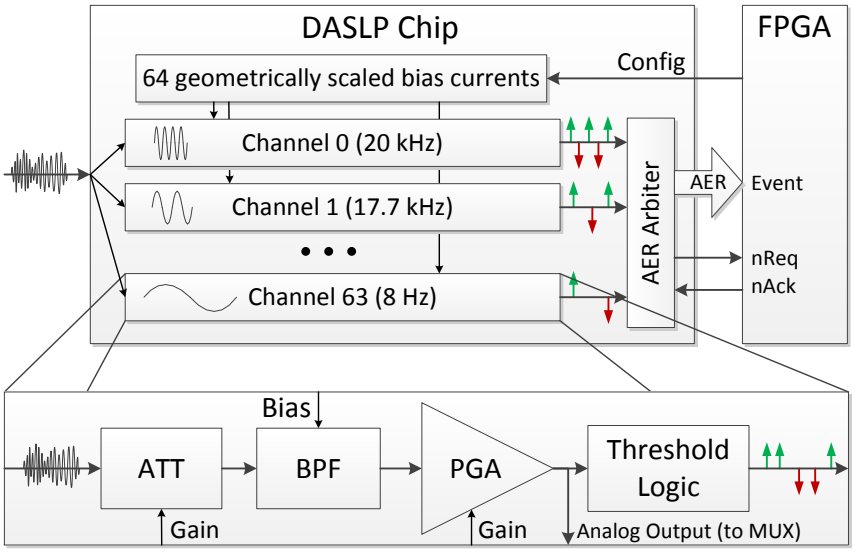

Figure 3.1: Architecture of the spiking cochlea DASLP. A simplified structure diagram of one channel is shown in the bottom callout. The green and red arrows represent ON and OFF spikes.

The filters model the functionality of the basilar membrane of the biological cochlea. Each filter channel produces both ON and OFF spikes unlike other DAS cochleas which do not produce the dual polarity spikes. The dual polarity spikes are produced by the send-on-delta scheme used for generating the asynchronous spikes. These spikes are transmitted off-chip through the asynchronous event representation (AER) protocol. The AER block arbitrates among all the active channels. The asynchronous handshaking signals C_{ack} and C_{req} are used to transmit the chosen channel address $Addr$ to the external device. The analog block operates on a power supply of 0.5 V and consumes only 55 μW. This design has good matching properties of the quality factor, Q, of the filter across the different channels, and $Q > 10$ can be achieved across the entire array (Yang, Chien, *et al.*, 2016).

The DASLP board (Fig. 3.2) that holds the chip is similar to the DAS board (Liu, Schaik, *et al.*, 2014), except that it uses more modern USB3 interface and has a bigger FPGA instead of a low-capacity CPLD for the chip control and data readout. It is USB powered and interfaces to our Java-based jAER software (*jAER Project* 2007) for setting the chip biases, recording, and processing sensor output. It is also equipped with a pair of on-board differential 18-bit ADCs (AD7691[1] from Analog Devices) that sample input signal to the left and right channels of the cochlea chip to allow direct comparison of the algorithms that use cochlea spikes versus the algorithms that use regular or other types of sampling [(Huber *et al.*, 2017, 2018)]. For the purpose of this work the input of the left channel ADC was rerouted to the analog output of the chip. The filter output of any channel can be internally multiplexed to this output, allowing measurement of the signal amplitude at the output of the filter. The data samples from both ADCs are sent to a PC along with the event data stream from the DASLP chip as described in the Appendix to the Chapter 3.

1 https://www.analog.com/media/en/technical-documentation/data-sheets/ad7691.pdf

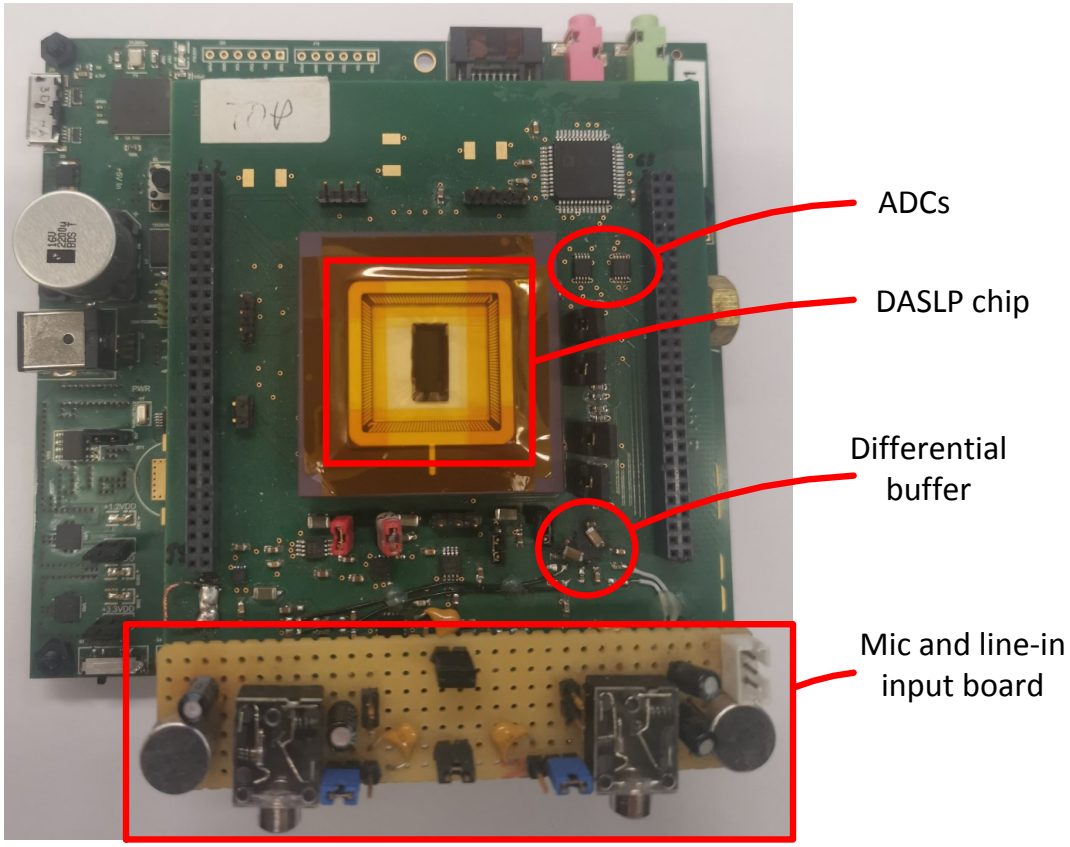

Figure 3.2: DASLP chip and board view.

Each channel has an individual programmable attenuator (ATT) and a programmable gain amplifier (PGA). There are 8 levels of attenuation ranging from 0 to -18 dB and 8 levels of gain ranging from 18 dB to 38.5 dB available (measured at the channel peak frequency with $Q = 1$). However, only 4 attenuation levels (-6 dB to -18 dB) were used in this work, because switching to the low attenuation range in this design was too slow to be used in a feedback loop (see Sec. 3.2.2 "Gain switching transient response", Fig. 3.5). By combining different attenuation and gain levels the overall gain range of $0 - 32.5$ dB can be achieved. Twelve levels of gain were available for the AGC control loop (Table 3.1). The combinations of the attenuator and PGA gains were chosen in such a way that each gain change step is around 3 dB (Fig. 3.3). Thirty-six channels were used in this work, corresponding to frequencies from 56 Hz to 4 kHz.

3.2.2 Gain Switching Transient Response

Loading a new gain and attenuation setting for every channel on the DASLP chip takes about 0.5 ms. It takes additional time for the output signal to settle down to the new amplitude level. We found that the time required for switching between the attenuator settings depends on the number of attenuation levels between the initial and final settings. In the case of rapid switching from setting #0 (-18 dB) to setting #7 (0 dB), the attenuation even changes in the opposite direction starting from the

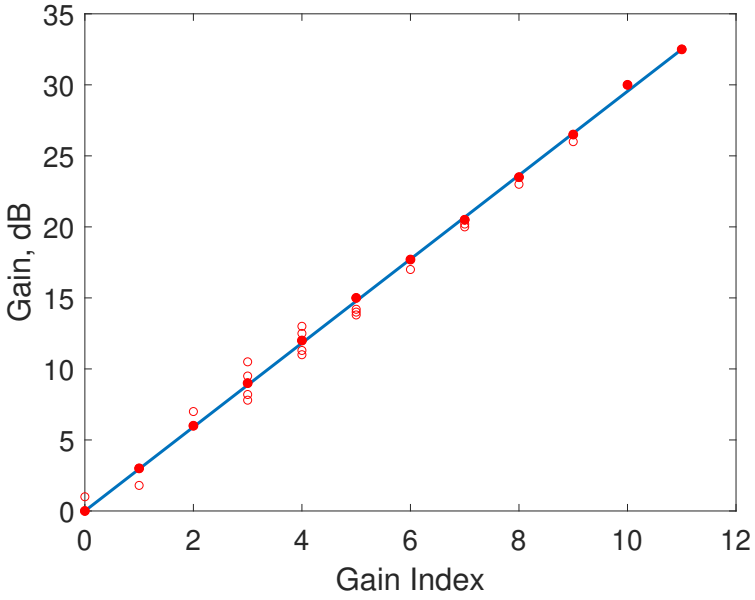

Figure 3.3: Selection of the gain values available for AGC. The selected gain values are marked with the filled circles, all other available gain values are shown with the empty circles.

Gain index (4 bit)	Total Gain, dB	Gain, dB		Bit Pattern	
		PGA	ATT	PGA	ATT
0	0	18	−18	111	000
1	3	21	−18	100	000
2	6	18	−12	111	001
3	9	21	−12	100	001
4	12	18	−6	111	011
5	15	21	−6	100	011
6	17.7	26.2	−8.5	011	010
7	20.5	29	−8.5	010	010
8	23.5	32	−8.5	001	010
9	26.5	38.5	−12	000	001
10	30	38.5	−8.5	000	010
11	32.5	38.5	−6	000	011
15	Threshold crossing special code				

Table 3.1: Gain settings available to AGC state machine and corresponding bit patterns for the PGA and the input attenuator. The last Gain Index value (15) is used to indicate a threshold crossing event for the corresponding channel (see Appendix A.2 Fig. A.11).

setting #4 (see the red circle in right plot of Fig. 3.4). Although the attenuation does settle at the selected level, the time constant of this process is in order of seconds. This effect is in part because the chip was not designed to support this fast attenuation switching. This behaviour can lead to the gain oscillations, when

the signal amplitude drops so low after increasing the attenuation, that the channel would stop generating any events for a substantial time, and the AGC would have to decrease the attenuation again. This problem is partially mitigated by using a restricted set of the attenuation values as shown in Fig. 3.5), however, we still observe such a behaviour in response to a sharp onset of a signal at the high frequency channels.

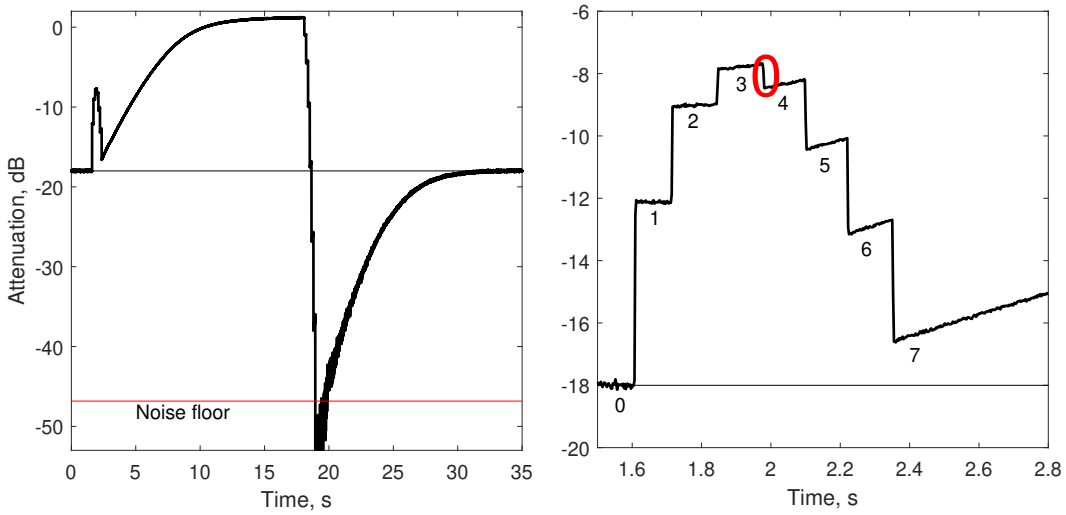

Figure 3.4: Stepwise switching between all attenuation settings from #0 to #7 and back. The right plot shows zoomed-in response to the #0-to-#7 attenuation setting stepwise transition. Actual attenuation value changes in the correct direction up to the setting #3 and then goes in a wrong direction (marked with the red circle) for a substantial time, making it impossible to use the settings #4 – #7 in a fast feedback loop.

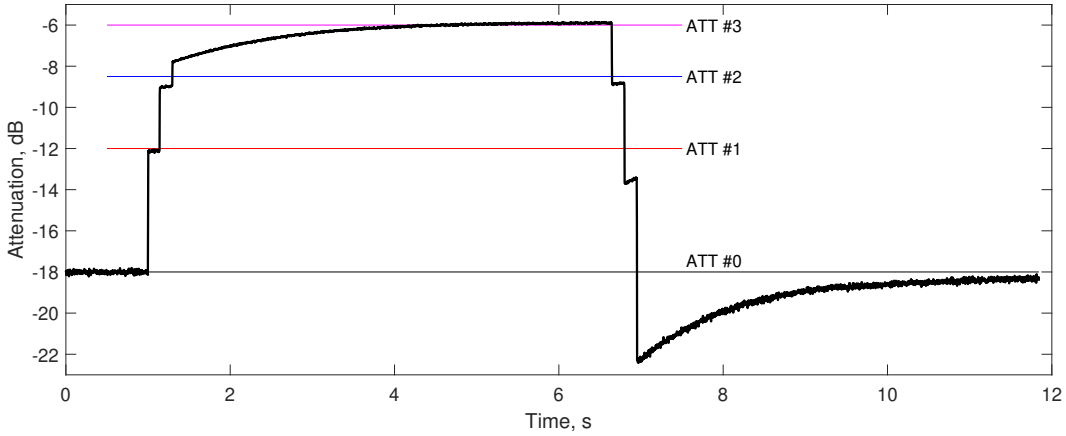

Figure 3.5: Stepwise switching between attenuation settings from #0 to #3 and back. When the attenuation setting is switched rapidly from #0 to #2 or #3, the actual attenuation value is below the specified level. Although this behavior may lead to unnecessary increase of the PGA gain, it does not cause complete signal loss. However, when the attenuation is switched in the opposite direction, the actual attenuation value overshoots the specified level significantly, that can cause temporary signal loss and hence, force the AGC to change the gain and the attenuation in the opposite direction.

3.2.3 Sensor Features

The DASLP implements an Asynchronous Delta Modulation (ADM) coding scheme for each frequency channel, which means ON and OFF events are generated when input signal exceeds positive or negative threshold with respect to the previously encoded level. This scheme preserves the information about both frequency and amplitude of the signal. The number of ON events on the rising slope of the signal encodes the signal amplitude, and the time interval between ON events on two consecutive rising slopes encodes the frequency (Fig. 3.6). Thus, the frequency of the signal at each channel can be estimated with high accuracy when averaged over several periods (Appendix A.2 Fig. A.18).

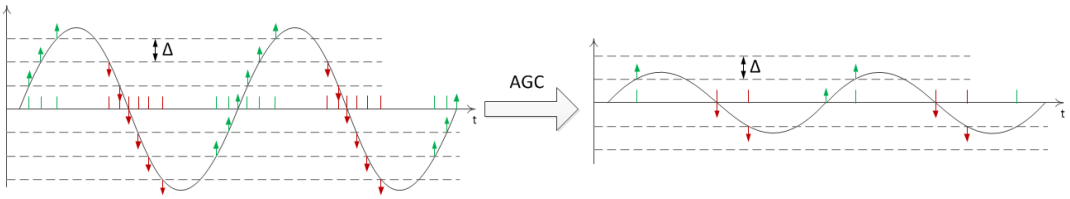

Figure 3.6: Spike generation with ADM

However, the output events of the DAS are often binned into time bins before sending them to the further processing steps, like a recurrent neural network classifier (Acharya *et al.*, 2018; Anumula, Neil, Li, *et al.*, 2017; Anumula, Neil, Delbruck, *et al.*, 2018; Gao, Braun, *et al.*, 2019; Yang, Yeh, *et al.*, 2018). This time-binning process discards the ordering information of ON and OFF events, leaving only total event counts per time bin per channel, which represents some mixture of the frequency and amplitude of the signal at the channel. Adjusting the gain of a channel in such a way that only one ON event per period is generated helps to keep the frequency and the amplitude information separated even when time-binning is applied. In this case the spike number in a time bin encodes signal frequency, and the gain of the channel during this time bin encodes signal amplitude.

3.2.4 Dataset Preparation

Speech samples were taken from the TIMIT dataset (Garofolo *et al.*, 1992) and noise samples from both, MS-SNSD (Reddy *et al.*, 2019) and MUSAN (Snyder *et al.*, 2015) datasets. Two hours of speech samples were taken from the TIMIT dataset (one hour from the training set, one hour from the test set) and two hours of noise samples consisting of environmental sounds and music were randomly selected from the MS-SNSD and MUSAN datasets. The noise dataset was randomly spilt into equal training and test sets, such that each audio file was included completely to either train or test set. Each audio file was normalised individually, so that the root mean square (RMS) amplitude of a signal at the input of the DASLP chip would be equal to the defined amplitude. The audio samples were played to the cochlea through a computer sound card. The schematic diagram for the input signal conditioning circuit is given in the Appendix A.2.1. The training set was recorded with amplitudes $[5, 10, 15, 50, 80]$ mV. Ten different amplitudes – $[2, 2.5, 5, 7, 10, 15, 20, 30, 50, 80]$ mV

were used for the test set recordings. The maximal gain setting available to AGC (32.5 dB) was used for the non-AGC recordings, so that there were a few spikes even at the smallest signal amplitude. At this gain level the input attenuation is set to −6 dB (Table 3.1), that reduces actual signal amplitude at the input of the PGA by factor of 2.

3.3 EVENT-DRIVEN AGC ALGORITHM AND HARDWARE IMPLEMENTATION

3.3.1 Spike-driven Local AGC Algorithm

The local AGC algorithm presented in this work is intended to maintain an average event rate of each channel within a certain range. The averaging time interval τ_{ch} is defined individually for each channel based on its best characteristic frequency F_{ch}. We use N periods of F_{ch} for computing a running estimate of the event rate r_{ch} of the channel, thus τ_{ch} is computed as follows: $\tau_{ch} = N/F_{ch}$. Since the DASLP channels are spaced approximately exponentially within the range of 8 Hz to 20 kHz, we use Eq. 3.1 to compute the channel characteristic frequency F_{ch}, and hence τ_{ch} (Eq. 3.2 and red curve in Fig. 3.7).

$$F_{ch} = 8 \cdot scaling\_factor^{(63-ch)} \quad (Hz) \tag{3.1}$$

$$\tau_{ch} = N/(8 \cdot scaling\_factor^{(63-ch)}) \quad (s) \tag{3.2}$$

The $scaling\_factor$ is computed by substituting $F_0 = 20000$ Hz and $ch = 0$ into equation 3.1, $scaling\_factor \approx 1.13224$.

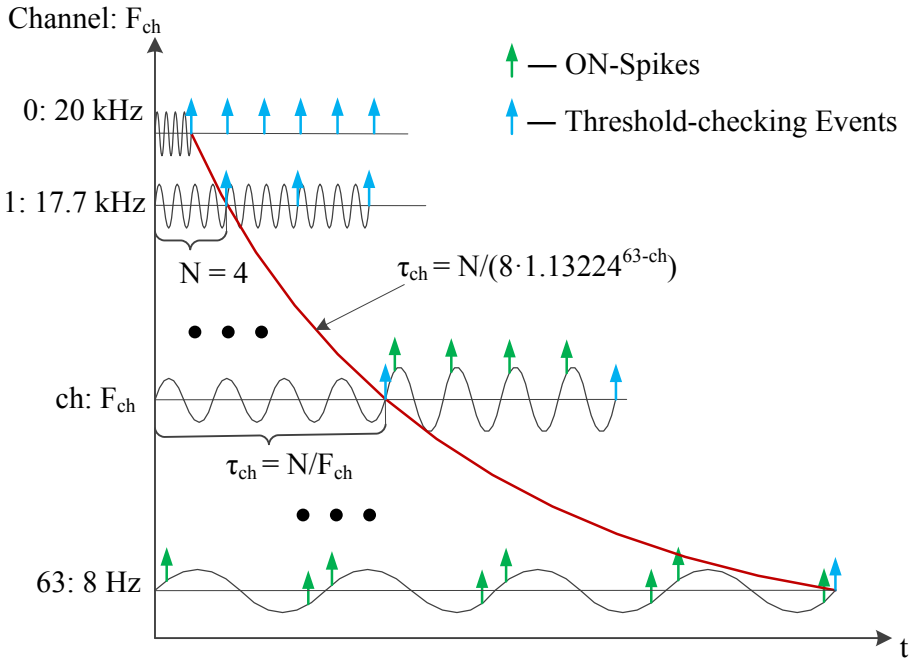

Figure 3.7: Spike rate averaging time intervals for different channels.

At the end of the averaging time interval (denoted by blue arrows in Fig. 3.7) the measured average event rate is compared to two programmable thresholds: the lower threshold T_l and the upper threshold T_u. These thresholds are the same for all the channels. When the measured average spike rate does not fall into the range defined by these thresholds, the AGC controller changes the channel gain to the next possible value in the direction opposite to the exceeded threshold and starts measuring the spike rate again. For example, in Fig. 3.7 the gain of the channel *ch* was increased because there were no spikes in the previous averaging time interval τ_{ch}. In order to simplify implementation of the proposed algorithm on an FPGA, we use spike counts sc_{ch} over N periods of F_{ch} instead of computing an average spike rate per F_{ch} period (r_{ch}) over N periods. We also specify the thresholds T_l and T_u as a number of spikes within the averaging time interval τ_{ch}.

In this work we use $N = 8$, the lower threshold $T_l = 1$ and the upper one $T_h = 16$, defining the acceptable range of spike rates as $0.125 - 2$ spikes per F_{ch} period on average.

The described algorithm is shown in a more formal form in Alg. 1. The AGC block diagram is shown in Fig. 3.9.

Algorithm 1: AGC control loop for one channel "*Ch*"

```
while True do
    while Time counter[Ch] < Averaging T[Ch] do
        if New event() and (Event address == Ch) then
            Event count[Ch] ++
        end
        Every 100μs Time counter[Ch] ++
    end
    if (Event count[Ch] ≥ Upper threshold) and (Gain index[Ch] > 0) then
        Gain index[Ch] −−
    end
    if (Event count[Ch] < Lower threshold) and (Gain index[Ch] < 11) then
        Gain index[Ch] ++
    end
    Current gain[Ch] = Gain lookup table[Gain index[Ch]]
    Time counter[Ch] = 0
    Event count[Ch] = 0
end
```

The spike rate averaging time window τ_{ch} also defines the AGC attack t_a and release t_r time for the channel. These time periods are defined as a time needed for the output signal to reach 90% of the change towards a new steady-state value after a step increase or decrease of the input signal amplitude, as shown in Fig. 3.8. With the proposed algorithm the Attack and Release times depend on the channel frequency and on the input signal amplitude change.

In the DASLP cochlea design each event carries information about channel number (6 bit) and polarity of the event (1 bit), which shows whether the event was generated at the rising or falling slope of a signal (Fig. 3.6). The threshold for event

generation is the same for all the channels, so the frequency components of the same amplitude would generate the same number of spikes per period at different channels. However, when local AGC is used, each channel can have different gain at different moments. The current gain value can be used for estimation of the amplitude of a frequency component corresponding to a channel in the input signal at the moment when the spike is generated at the channel. So, we embed the current gain setting for the channel that generated an event into this event. Since we have 12 gain settings, 4 bit are required to carry this information. Thus, each event carries 4 bit more information when the local AGC is enabled. The bit-structure of the DASLP event is given in the Appendix A.2 Fig. A.10.

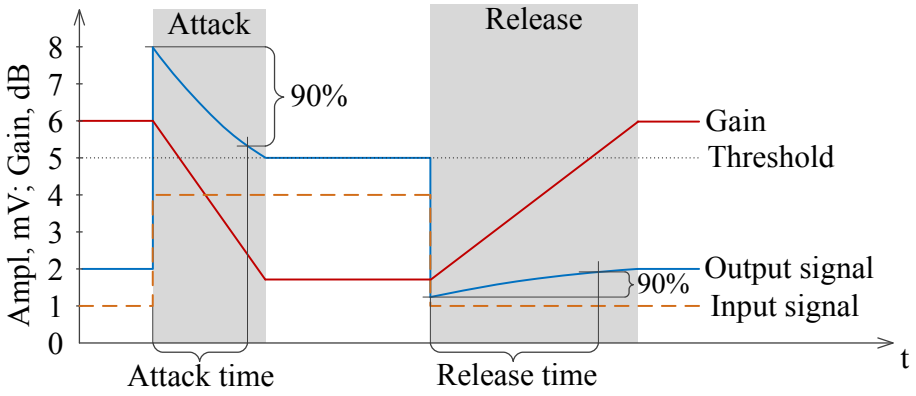

Figure 3.8: Definition of AGC Attack and Release time.

According to the given definition and assuming maximal gain change from 32.5 dB to 0 dB, the 90% of the amplitude change is reached between gain values 15 dB and 12.5 dB (Table 3.1), which results in 7 gain change steps for the attack time: $t_a = 7 \cdot \tau_{ch}$. The maximal release time achieved in the case of gain changing from 0 dB to 32.5 dB and equal to $t_r = 11 \cdot \tau_{ch}$, because about 25% of the total amplitude change happens at the last step (from 30 dB to 32.5 dB).

3.3.2 FPGA AGC Implementation

Our FPGA implementation of the described AGC algorithm is completely counting based and it uses integer arithmetic only, so it does not require any multipliers, dividers or any other DSP resources of the FPGA (Table 3.2).

There are four memory registers associated with each AGC channel:

- 12-bit register for storing the length of the averaging window for a channel in 0.1 ms steps;
- 12-bit register for a counter which represents the time that passed from the beginning of the current averaging window in 0.1 ms steps;
- 6-bit event counter;
- 4-bit current gain index for a channel.

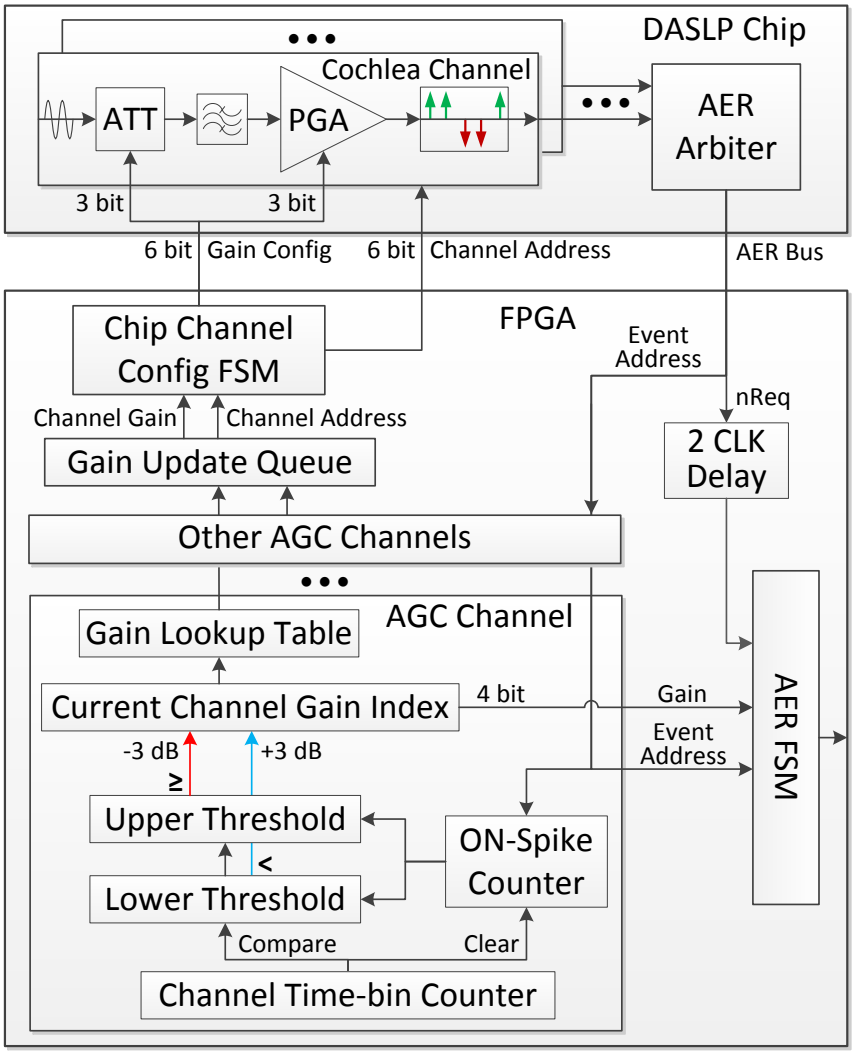

Figure 3.9: Block diagram for FPGA implementation of AGC algorithm.

In addition to these registers two memory bits per channel are used:

- Channel Enable bit – enables the AGC operation at the channel;
- Averaging Window End bit – indicates that the averaging time passed and the channel event count has to be compared with the thresholds.

Thus, 36 bits of memory are used for one AGC channel. There is also a lookup table that translates 4-bit channel gain index into 6-bit gain setting bit pattern specified in Table 3.1. It is shared between all the channels.

By setting the time window counter resolution to 0.1 ms and using 12-bit registers, the length of the averaging time window can be selected in a range from 0.1 ms to 409.5 ms in steps of 0.1 ms. The length of the time window is programmed separately for each channel from jAER software. The 6-bit event counter saturates at the value 63, however it does not overflow – when it reaches its maximal value, it stops counting and keeps this value until the end of the channel averaging time window. At the end of this window, the event counter is compared to two

thresholds. If the spike count is less than the lower threshold, the "gain increase" event is generated, and if the spike count is greater or equal to the upper threshold, the "gain decrease" event is generated. See Appendix A.2 Fig. A.11 for the gain update event bit-structure.

When either of the gain update events is generated, it is injected into the AER output FIFO immediately, however updating the gain settings of a channel in the DASLP cochlea chip takes 0.5 ms, and several gain update requests can occur at the same time or during the time when the system is updating the configuration of the current channel. This would lead to missing of the gain updates for some channels. In order to avoid this problem we had to implement a queue for the gain update requests. The queue is implemented as a 128 $\times$ 12-bit FIFO, where 6 bits represent the channel address and the other 6 bits represent the bit pattern for the new gain setting. The FPGA resources occupied by the 64-channel AGC controller are shown in Table 3.2.

	LUT4	**LUTRAM**	**FF**	**BRAM** (18Kb)	**DSP**
Available	66528	6804	49896	240	128
Used	5007	48	3217	1	0
Percentage	7.5%	0.7%	6.4%	0.4%	0%

Table 3.2: Resource utilization of AGC control logic on FPGA LATTICE LFE3-70EA.

3.4 RESULTS

This section presents experimental measurements of the cochlea responses in the presence and absence of the proposed local automatic gain control (AGC) mechanism. We first show in Sec. 3.4.1 the dependence of the filter channel output amplitude over a range of input frequencies and how this frequency response varies a range of input amplitudes. The effect of AGC on the steady-state spike responses are then presented in Sec. 3.4.2.

3.4.1 Dependence of Analog Filter Output on Input Amplitude

A big reason why AGC is needed for the DASLP spiking cochlea is that first, the chip analog core power supply is just 0.5 V, therefore the analog signal voltage at any part of the circuit can never exceed this value. Second, to ensure that the transistors circuits operate in the region needed to implement the intended filter transfer function (Yang, Chien, *et al.*, 2016), there is a constraint on the amount of amplitude change ($\approx$ 10 mV) of the input arriving at any filter, that is, in the case of a sine wave input, the Root Mean Square (RMS) value should be approximately smaller than 2.5 mV. Note that the filter circuits have differential inputs and outputs.

The output amplitude of the analog filters should be high enough to produce spikes at the spike generating circuit, therefore if the input amplitude is too low, the output amplitude might not be high enough to generate output events from the channel even though the frequency of the input is within the passband of the filter.

AGC is then useful to amplify the input amplitude to a range so that events will be generated. In contrast if the input amplitude is too large, the transistor circuits no longer implement the filter transfer function, therefore AGC is useful in bringing down the input amplitudes to the proper range.

In order to estimate the input amplitude range of the filters for linear operation in the DASLP filter bank we measured the frequency response of several channels at different gain settings and input amplitudes. The signals of different frequencies were played from a PC to the DASLP cochlea through a sound card. The output of the sound card was calibrated such that the frequency response was flat at the input of the DASLP chip, as shown in Appendix A.2.2.

We sampled the input signal to the DASLP chip and the output signal of six bandpass filters (channels 48, 42, 36, 30, 23, 13) with two differential onboard ADCs described in the Section 3.2.1 at 44.1 kHz sampling rate. We used a set of 100 exponentially scaled frequencies for measuring the frequency response of each channel. The frequency ranges are given in Table 3.3.

Ch #	F_{ch}, Hz	F_m, Hz	Freq sweep, Hz
13	3980	4550	400 – 8000
23	1150	1500	200 – 4000
30	482	640	100 – 2000
36	229	290	40 – 1000
42	109	133	40 – 1000
48	52	58	40 – 1000

Table 3.3: Parameters of the channels used for frequency response measurement. F_{ch} — designed channel frequency, F_m — measured peak frequency, Freq sweep — range of exponentially spaced frequencies used for frequency response measurements.

The frequency response is computed as a ratio of the output and the input RMS amplitudes. However, direct measurement of the output signal amplitude is not possible at low amplitudes due to high noise level at high PGA gain and high Q value (see Fig. 18 in Yang, Chien, *et al.*, 2016). The noise RMS amplitude at the highest PGA gain setting (38.5 dB) and $Q \approx 4$ is around 4.5 mV which is on par with the signal level at low amplitudes. In order to account for the noise in the measured output signal, we measured the noise level of each channel at each gain setting and subtracted the corresponding value from the measured signal RMS amplitude using the following equation:

$$A_{na} = \sqrt{A_{out}^2 - N(g)^2} \tag{3.3}$$

Where A_{out} is the measured RMS amplitude of the signal at the output of the filter, $N(g)$ is the measured noise level for the gain setting g, and A_{na} is the signal amplitude adjusted for noise. An example of the noise recording for different gain settings with computed RMS amplitudes of the noise is given in Fig. 3.10.

Thus, the gain of a channel is computed as follows:

$$G = 20 * lg(A_{na}/A_{in}) \tag{3.4}$$

Where A_{in} is the RMS amplitude of the signal at the input of the DASLP chip.

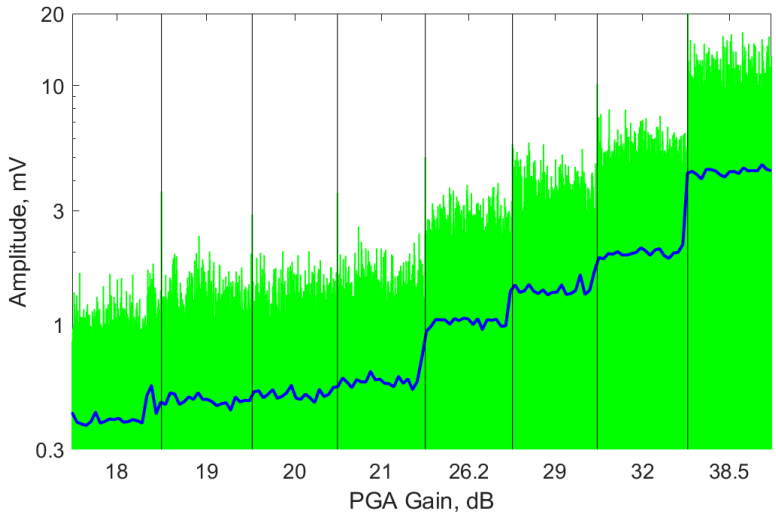

Figure 3.10: Noise samples recorded at the output of the channel 30 at different PGA gains (green). The blue line shows the RMS amplitude of the noise computed in 0.2 s bins. The average RMS amplitudes of the noise for each gain value are 0.4, 0.48, 0.52, 0.58, 1.04, 1.37, 1.98 and 4.42 mV, correspondingly.

The designed frequencies of the selected channels defined by Eq. 3.1 in Section 3.3 are 52, 109, 229, 482, 1150 and 3980 Hz correspondingly. However, the peaks of the frequency responses measured at low amplitudes are shifted towards higher frequencies as shown in Fig. 3.11. The measured values of the characteristic frequencies are 58, 133, 290, 640, 1500 and 4550 Hz, correspondingly (Table 3.3).

The frequency responses for six channels are plotted in Fig. 3.11. The curves are obtained for five different RMS amplitudes of 1, 3.5, 7, 10 and 14.2 mV. The transfer functions from all six channels obtained with the lowest input amplitudes of 1 and 3.5 mV overlap significantly and has the highest measured gain at 32.5 dB and above. The curves also show that because of transistor mismatch, the peak gains for different channels vary between 32.5 dB and 38.5 dB. For input amplitudes higher than approximately 7 mV RMS, the shape of the filter function changes because the transistor circuits no longer implement the intended linear filter transfer function. We see that the peak of the transfer function and the Q value decreases for increasing amplitudes at every channel.

Fig. 3.12 shows the filter output of channel 23 as a function of frequency for different input amplitudes. The gain setting of the filter is set at 32.5 dB. The measurements show that an output amplitude of 23 mV and above is needed for spikes to be produced. The output amplitude has to exceed 23 mV for spikes to be generated at the channel. But as the input amplitude increases, spikes are generated even at frequencies that are far away from the best center frequency because the transfer function becomes non-linear when the input amplitude increases past 5 mV.

Fig. 3.13 shows the measurements again from channel 23 but at a lower gain setting (around 9 dB). The curves at this lower gain setting for the same range of input amplitudes show the transfer function stays linear for input amplitudes up to 20mV. As the input amplitude increases, the peak of the frequency response curves

start shifting towards lower frequencies indicating that the transistor circuits are no longer linear. When the output signal approaches 300 mV (Fig. 3.13, blue plot with square marks), the frequency peak is shifted by $\sim$ 25%. The bottom plot of the accompanying gain values versus frequency shows a similar trend of the best frequency moving towards the left as the input amplitude increases. It also shows that the peak gain only changes by 1 dB across all input amplitudes.

Fig. 3.14 shows an example of the analog output of a channel when the circuit is no longer linear. The output becomes a distorted form of the input sine wave. The corresponding ON and OFF spikes are shown in the bottom-most curve.

Fig. 3.15 shows the non-idealities in the measured output amplitudes at a different channel (channel 30). We can see that for the range of presented inputs, the corresponding output amplitudes again start saturating reaching a limit of 300 mV. The equivalent gain curves are shown in Fig. 3.16. The measured gain curves in Fig. 3.17 show that they can be maintained over a larger range of input amplitudes when AGC is enabled.

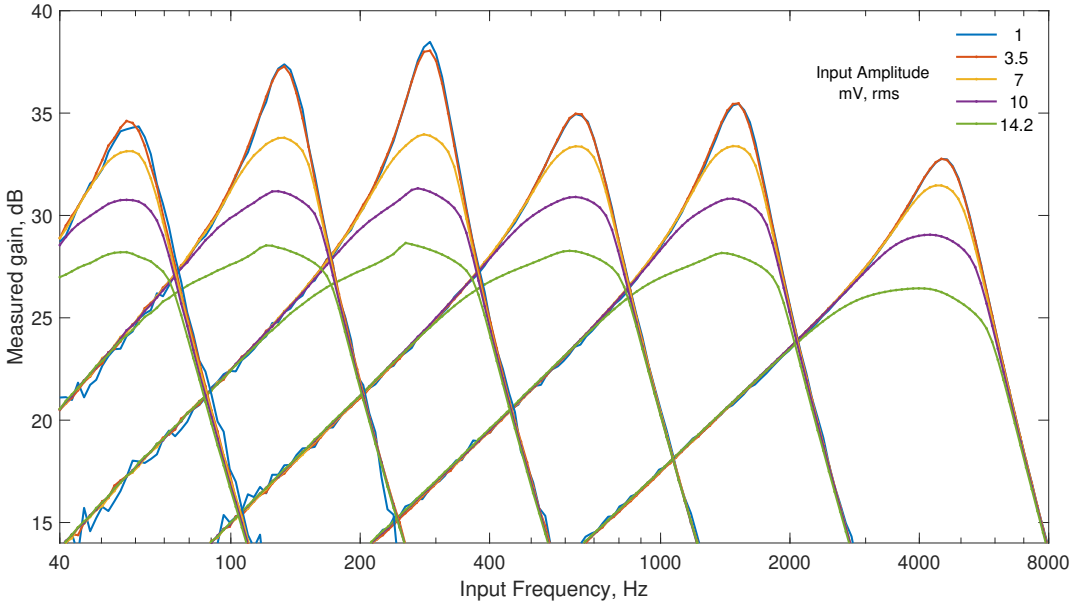

Figure 3.11: Frequency response of channels 48, 42, 36, 30, 23 and 13 (58, 133, 290, 640, 1500 and 4550 Hz, correspondingly) at different input amplitudes. ATT $=$ -6 dB, PGA gain is set to 38.5 dB. Total gain is 32.5 dB. Actual gain measurements for small signal amplitudes show that the total gain varies from 32.5 dB at the channel 13 to 38.5 dB at the channel 36 because of mismatch.

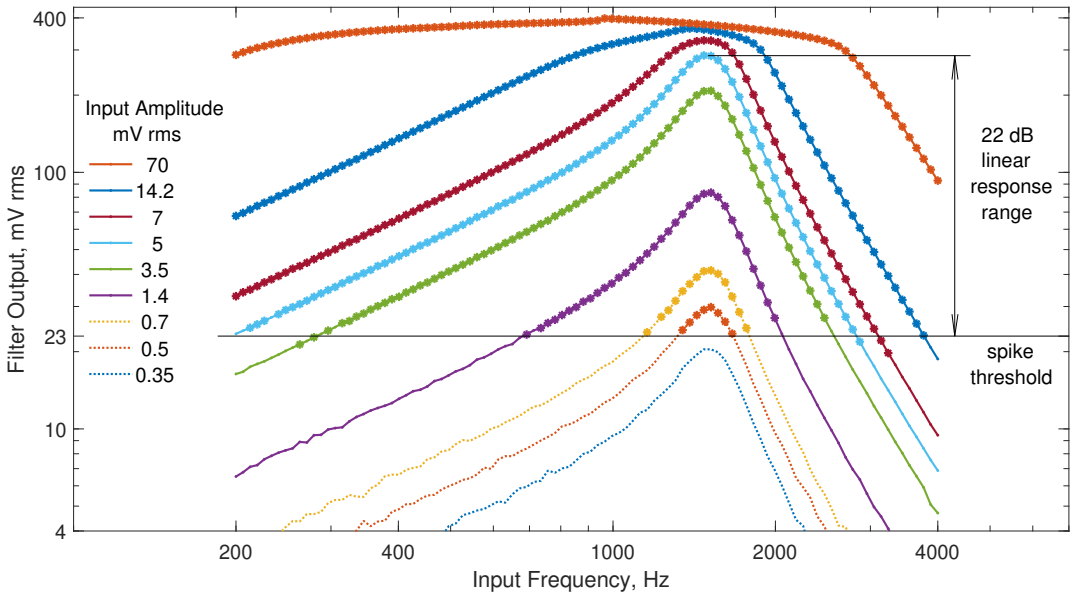

Figure 3.12: Amplitude frequency response of channel 23 at different input amplitudes for total gain setting of 32.5 dB. Larger filled dots indicate points where spikes were obtained.

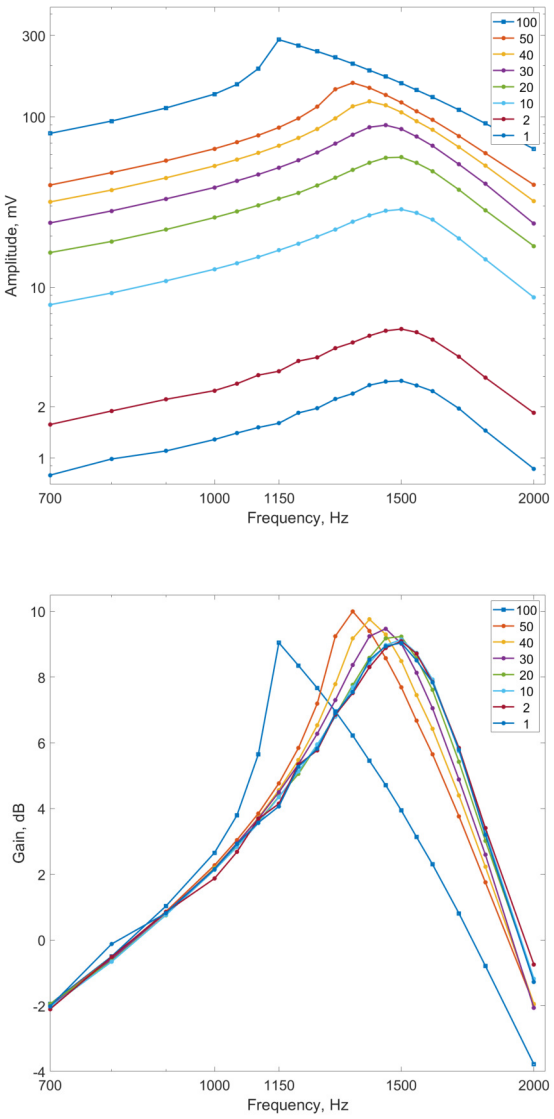

Figure 3.13: Output amplitude and gain versus frequency measured at channel 23. The different curves correspond to measurements for different input amplitudes (mV), Q = 4. Gain setting = 9 dB.

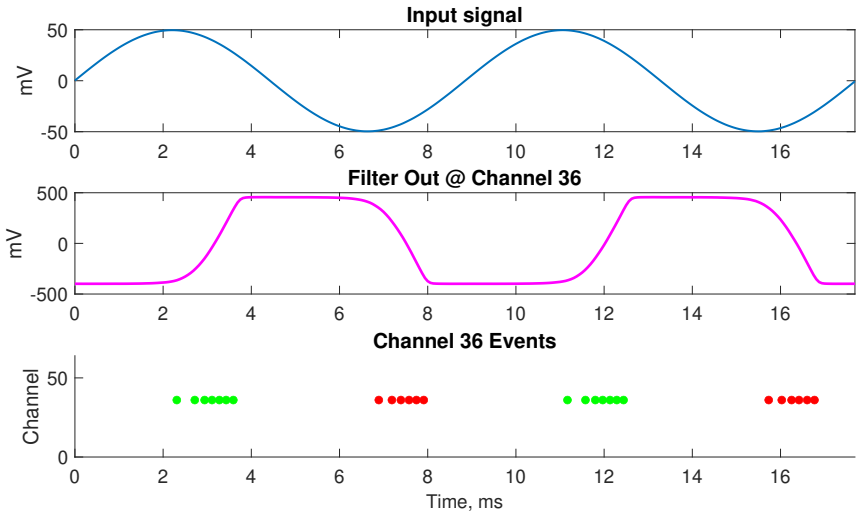

Figure 3.14: Measured outputs of channel 48 spike response. Input RMS amplitude is 35 mV (50 mV p-p). Output amplitude is 430 mV p-p. Bottom most trace shows the ON spikes (green) and OFF spikes (red) generated from the channel.

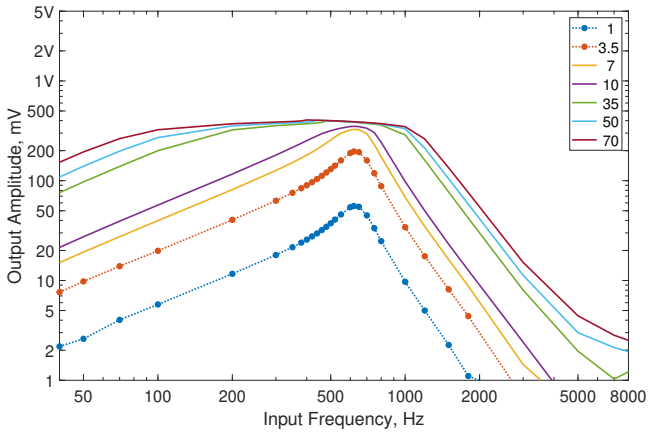

Figure 3.15: Amplitude frequency response of channel 30 at different input RMS amplitudes (see legend) with maximal gain setting (−6 dB ATT + 38.5 dB PGA = 32.5 dB total gain).

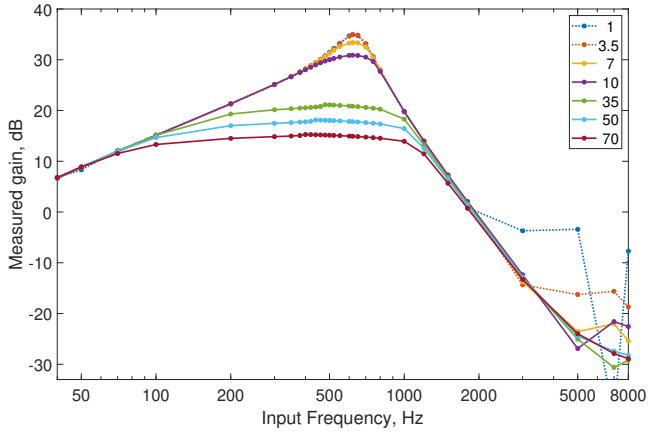

Figure 3.16: Gain frequency response of channel 30 for different input RMS amplitudes. Units in mV.

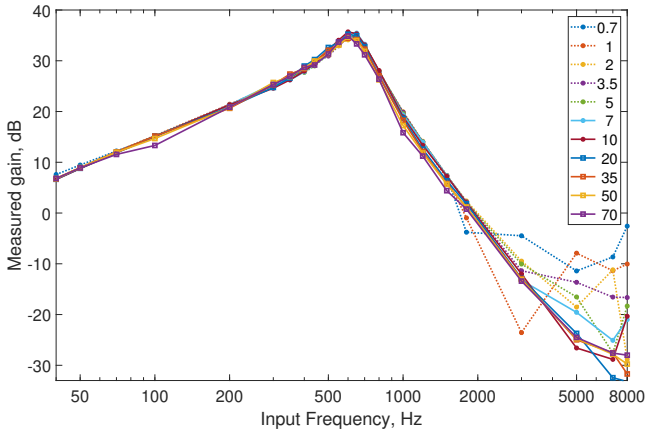

Figure 3.17: Gain frequency response of channel 30 when AGC is enabled. Curves correspond to different input RMS amplitudes. Unit in mV.

3.4.2 AGC Steady-state Spike Response Measurements

To demonstrate the change in the spike rates from the local AGC in response to an input signal amplitude change, the input signal and the analog output of the bandpass filter of one channel (#30, 642 Hz) is plotted in Fig. 3.23 (top) along with the channel spike response to this signal (bottom). The gain change events (red dots) shown in the lower plot of the figure indicate gain setting changes when the spike count in the time window has exceeded one of the thresholds. The ordinate of the dot shows a new gain value for the selected channel. When the channel gain has its extreme values (0 or 32.5 dB) it does not change further in the corresponding direction, thus one can see two gain change events at the same gain level in the beginning of the plot. The blue dots indicate the cochlea ON-spikes at the selected channel, their ordinate represents the current gain of the channel at the time when the event was generated.

The plot shows that at the beginning, the channel produces no spikes because there is no input signal, therefore the gain stays at the highest setting. When the speech sample starts, the output amplitude is initially high because of the high gain. Over a period of about 150 ms the gain value decreases in steps until the spike count (yellow bars in Fig. 3.23) drops below two spikes per period. The spike count represents the average number of the cochlea spikes per period of the channel characteristic frequency within the averaging window. The width of the bars is proportional to the length of the averaging time window. The gain setting stays constant at 3 dB during the period of $0.2 - 0.5$ s, when the spike rate fits within the desirable range, and then gradually increases close to the maximal value during the period of $0.56 - 0.68$ s, when no spikes are produced at the channel 30.

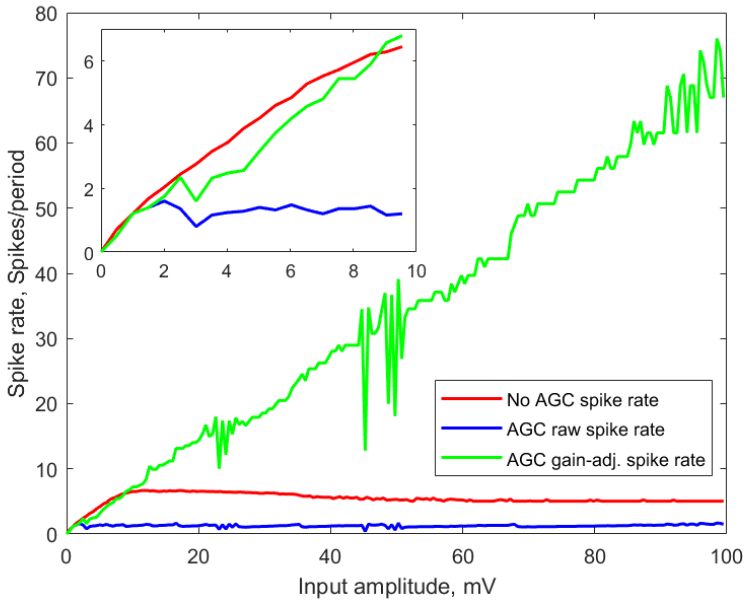

Figure 3.18: Frequency-normalized spike rate of channel 30 ON-spikes for three cases: without AGC, raw spike rate with AGC, and gain-adjusted spike rate with AGC. The inset in the left top corner shows a zoom in of the same data for the input range $0 - 10$ mV.

Figure 3.18 shows the spike rate of a channel over a range of input amplitudes from 0 mV to 100 mV when the AGC mode is enabled or disabled. A sine wave (F = 500 Hz) with increasing amplitude was applied to the input. When the cochlea operates in the non-AGC mode (red trace), the spike rate first increases with the input amplitude, but then it quickly saturates when the filter goes out of small signal operation. So, it is not possible to reconstruct the amplitude of the input signal based on the spikes from one channel. When the AGC is enabled (blue trace), the spike rate stays approximately constant over 40 dB range of input amplitude (1 mV to 100 mV). However, there is an additional information transmitted with the event — the current gain, G_{ch}, of a channel. We can use this information to estimate an amplitude of the input signal more accurately. Higher the channel gain G_{ch}, lower the input signal amplitude A_{in} needed to produce the same spike rate r_{ch} at the channel, so in order to estimate input amplitude we need to divide the spike rate by the gain of the channel (defined in terms of the output and input amplitudes ratio).

$$\hat{A}_{in} \propto r_{ch} / 10^{G_{ch}/20} = r_{ch} \cdot 10^{-G_{ch}/20} \tag{3.5}$$

where $\hat{A}_{in}$ is the estimated input amplitude and G_{ch} is the gain of the channel measured in dB.

For the non-AGC plot in Fig. 3.18 the channel gain is set to the maximal value available to AGC $G_{max} = 32.5\,\text{dB}$, so in order to compare AGC and non-AGC cases we use a coefficient $10^{G_{max}/20}$ to equate the estimated spike rate for non-AGC and AGC cases at small input amplitudes, when the AGC also uses the highest gain:

$$r_{ga} = 10^{G_{max}/20} \cdot r_{ch} \cdot 10^{-G_{ch}/20} = r_{ch} \cdot 10^{(32.5-G_{ch})/20} \tag{3.6}$$

The results show that the gain-adjusted spike rate r_{ga} (green trace) is linearly proportional to the input amplitude in a wide range (40 dB) of input amplitudes. The spiky glitches in the response are due to transient processes during switching between some of the gain settings.

3.4.3 Spike Frequency Responses for non-AGC and AGC Cases

In this section we study a possibility of single frequency detection based on channel spike rate for non-AGC and AGC cases. It is clear from the description of the Send-on-Delta sampling in Section 3.2.3 (Fig. 3.6) that the spike rate of a channel does not represent the frequency of the input signal directly. Instead, it is proportional to both, frequency and amplitude of the signal. Such an entanglement of frequency and amplitude makes it difficult to use simple features, such as time-binned spike counts of interspike intervals distribution, for training classifiers or other models to be invariant to voice pitch or amplitude. We test a hypothesis that using local AGC may help to disentangle these parameters – if we continuously tune the gain of each channel such that only one spike per period is generated on average by the channel, then the spike rate would represent the signal frequency and the gain of the channel would be inversely proportional to the signal amplitude.

We played a set of 22 single tone sine waves with frequencies ranging from 40 Hz to 8 kHz spaced approximately exponentially (40, 50, 70, 100, 150, 200, 300, 400, 500, 600, 800, 1000, 1200, 1500, 2000, 2500, 3000, 4000, 5000, 6000, 7000, 8000 Hz) at 6 amplitudes from 2 mV to 70 mV and recorded spike responses for every 3rd channel starting from channel 0 for a total of 19 channels.

Each frequency was recorded for 9 sec. The mean and Standard Deviation (STD) of the instantaneous firing rate (IFR) for each channel were computed over this period. The IFR based on ON-spikes and its STD computed from the raw spikes in the non-AGC case are displayed in Fig. 3.19. The mean spike rate correspond to the mean of the IFRs. The six subplots correspond to the six input amplitude levels. Each subplot has 19 response curves for the 19 channels. The frequencies played are depicted as thin black horizontal lines, the instantaneous firing rates are shown with color lines and the color represents the frequency that was played. The shadowed regions are the standard deviations of the IFR. The IFRs of the raw spikes are higher than the input frequencies because multiple spikes are produced per cycle.

In order to estimate the input signal frequency by measuring interspike intervals we implemented a spike filtering method, which leaves only the first spike of the ON spike train and discards all subsequent ON spikes at the channel until the first OFF spike is received. After such a filtering, only first ON spikes corresponding to the signal rising slope of each period remain, leaving just one spike per input signal period, and hence, making possible estimation of the signal frequency by averaging IFRs of the filtered spike train. We see in Fig. 3.20 that the IFR now corresponds to the played input frequencies (all color lines are aligned with the corresponding black lines).

The plots show that as the input amplitude increases, more channels respond even if the frequency is not within the passband of the channels. For a channel that responds to a selected frequency at the lowest amplitude, the spiking rate of this channel to the same frequency increases as expected.

When AGC is enabled, the IFR responses as computed from the raw spikes for the 6 amplitudes are shown in the individual subplots of Fig. 3.21. The four high frequency channels (0, 3, 6, 9) were excluded from the AGC algorithm because the averaging windows for computing spike rates for these channels are too small

(see 3.2 in Section 3.3.1. This would lead to a large number of gain updates, which take 0.5 ms each, slowing down AGC response at all other channels.

The outcome of the AGC is that the maximum mean spike rate stays approximately below 80 kHz across the input amplitudes from 10 to 70 mV. There is a higher variance in the computed mean spike rates because when the AGC enabled because of instability in the AGC loop. The spike rates do not remain constant even for a single frequency steady-state condition.

The corresponding IFR values computed from the filtered spikes are similar to that obtained in the non-AGC condition that shown in Fig. 3.20, as seen in Fig. 3.22, that is, the IFR corresponds to the played input frequencies. Additionally, when comparing the resulting transfer functions with those of the non-AGC condition (Fig. 3.20), we see that the number of channels that respond to a particular frequency has decreased especially for the higher input amplitudes.

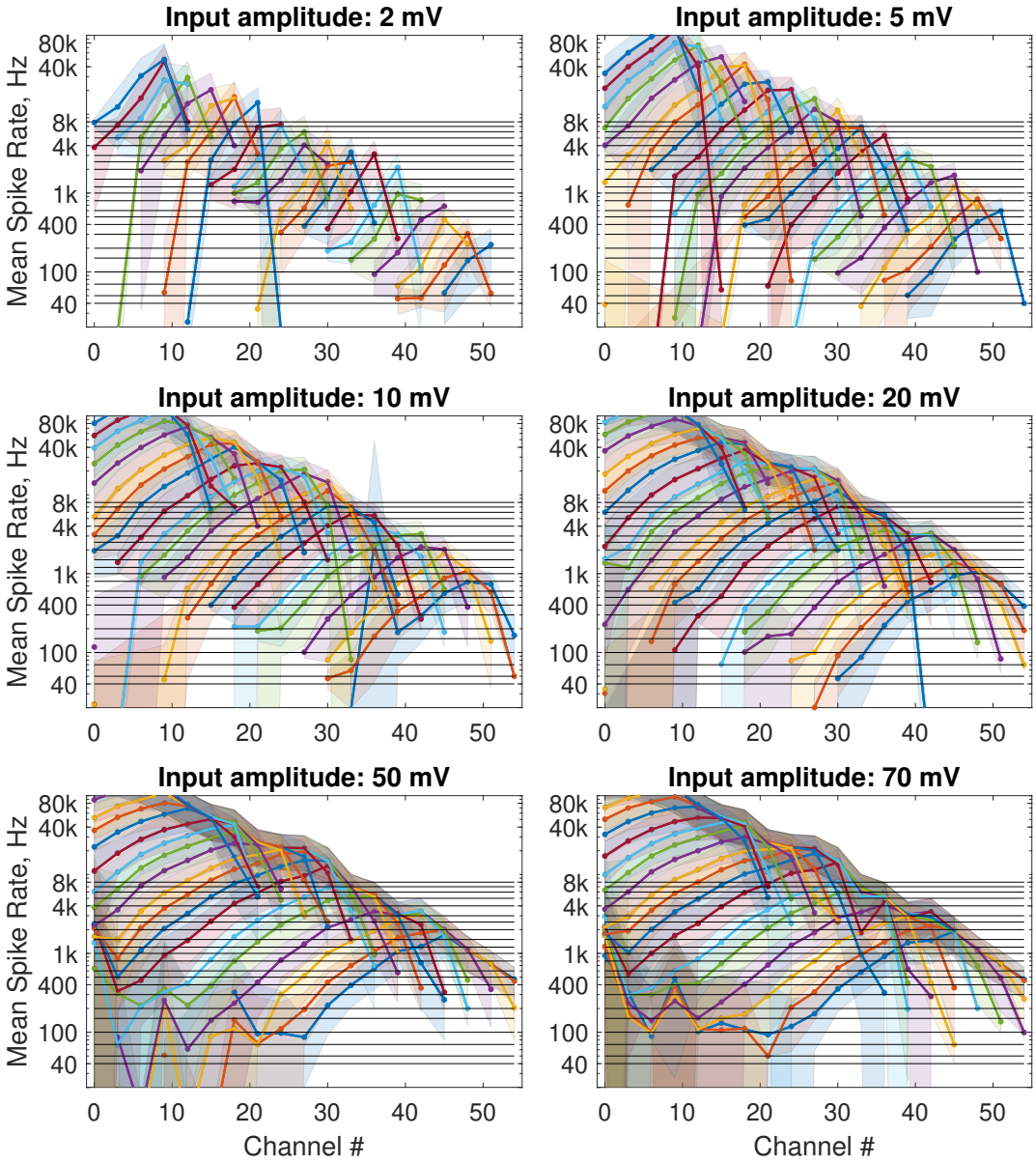

Figure 3.19: Spike frequency response obtained from the raw ON spikes in the non-AGC case. Gain setting = 32.5 dB. The mean spike rate is computed as a mean of inverse of interspike intervals. Each color line shows responses of all the channels to a single input frequency. The matching between the colors and the frequencies is the same for Figures 3.19 – 3.22. It can be inferred from Fig. 3.20 where the color lines are lined up with the thin black lines, which show all played frequencies. Shadowed regions represent the standard deviation (STD) of the spike rates. The plots show that even at the smallest input amplitude the spike rate does not represent the frequency played to the cochlea.

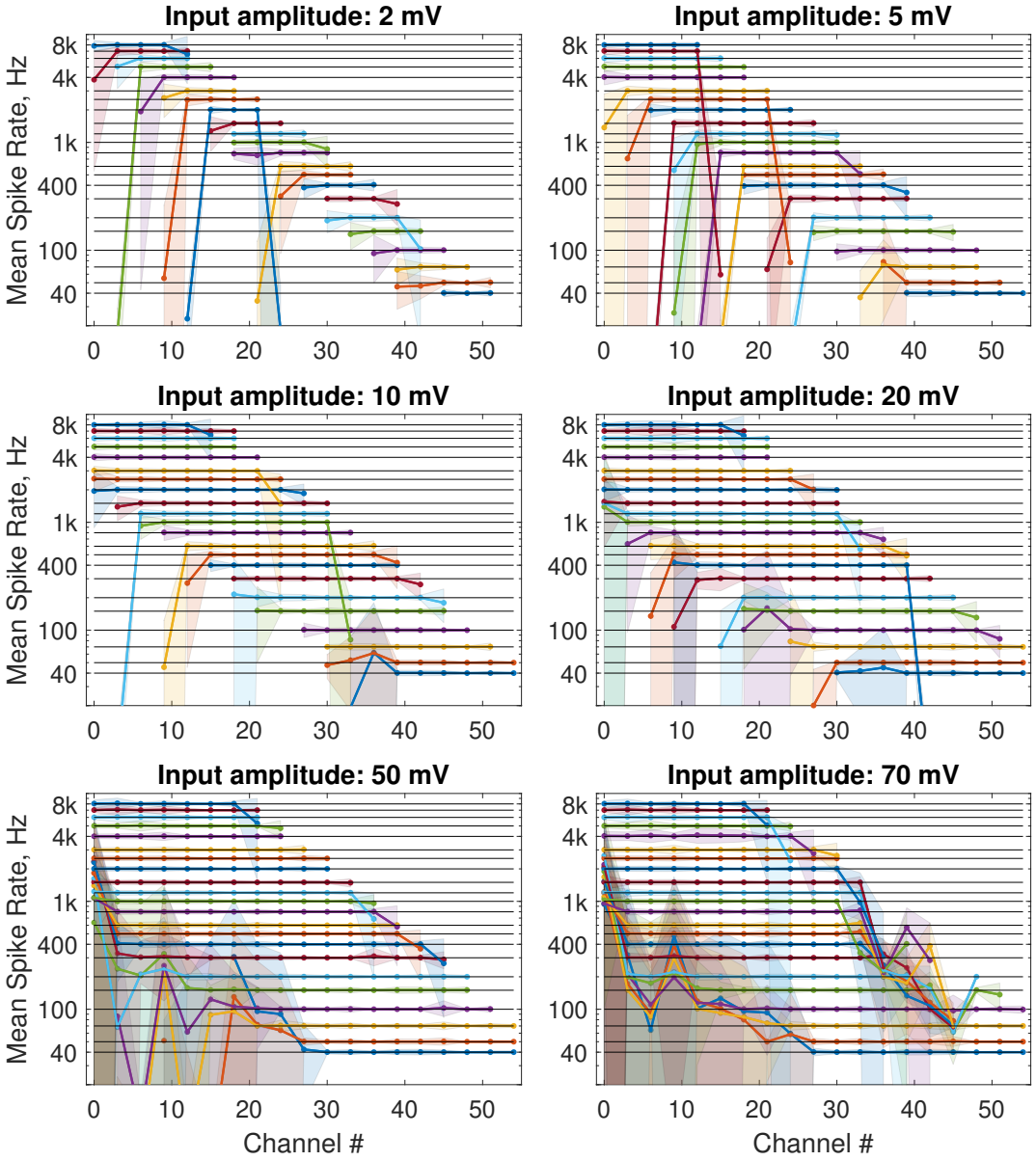

Figure 3.20: Spike frequency response obtained from filtered ON spikes in the non-AGC case. Gain setting = 32.5 dB. Filtering is done by keeping only the 1st spike in a series of ON spikes and discarding all subsequent ON spikes until at least one OFF spike is encountered. The resulting ON spike series is used for computing mean spike rate and its standard deviation (shown with shadowed regions). The plots show that after described filtering mean spike rate can be used for estimation of the frequency of a signal, however, at high amplitudes high frequency channels respond inadequately to low frequency signals (see left bottom corners of 50 and 70 mV plots.) This may be caused by the fact that the filters start to oscillate at their characteristic frequencies in a presence of a high amplitude out-of-band signal. These oscillations are suppressed to some extent by the AGC, as shown in Fig. 3.22.

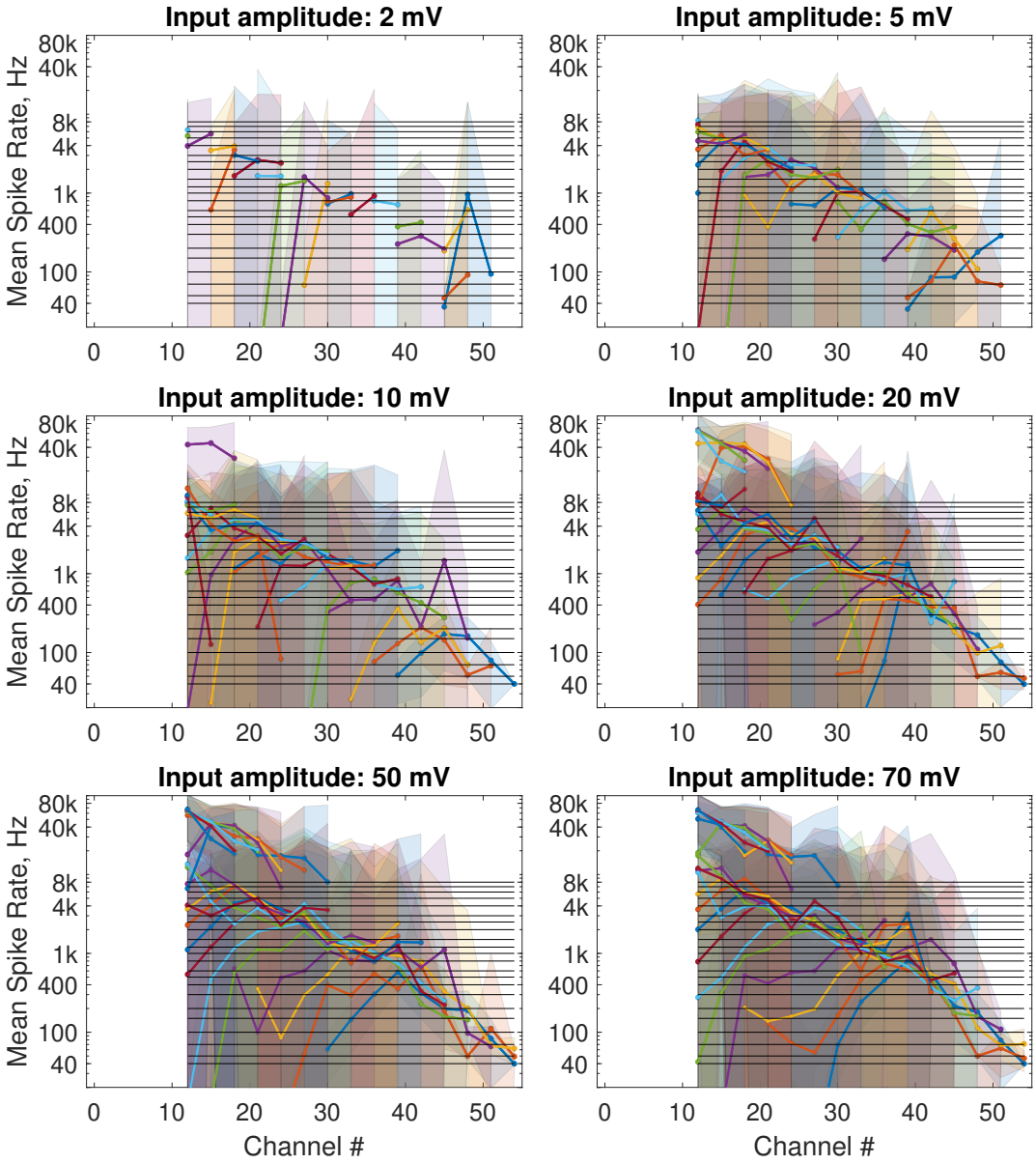

Figure 3.21: Spike frequency response obtained from raw ON spikes with AGC. These plots are analogous to the plots in Fig. 3.19, but the local AGC is enabled. The high frequency channels up to channel #13 are not measured because the gain update frequency for these channels would be too high. The measured mean spike rate is lower than in the non-AGC case, however the STD is higher as displayed by the shaded regions. This is due to the fact that AGC can switch the gain of the channel because of the noise at the channel output, even when the input signal is stationary. The plots show that the frequency estimation based on the spike rate averaging is still infeasible.

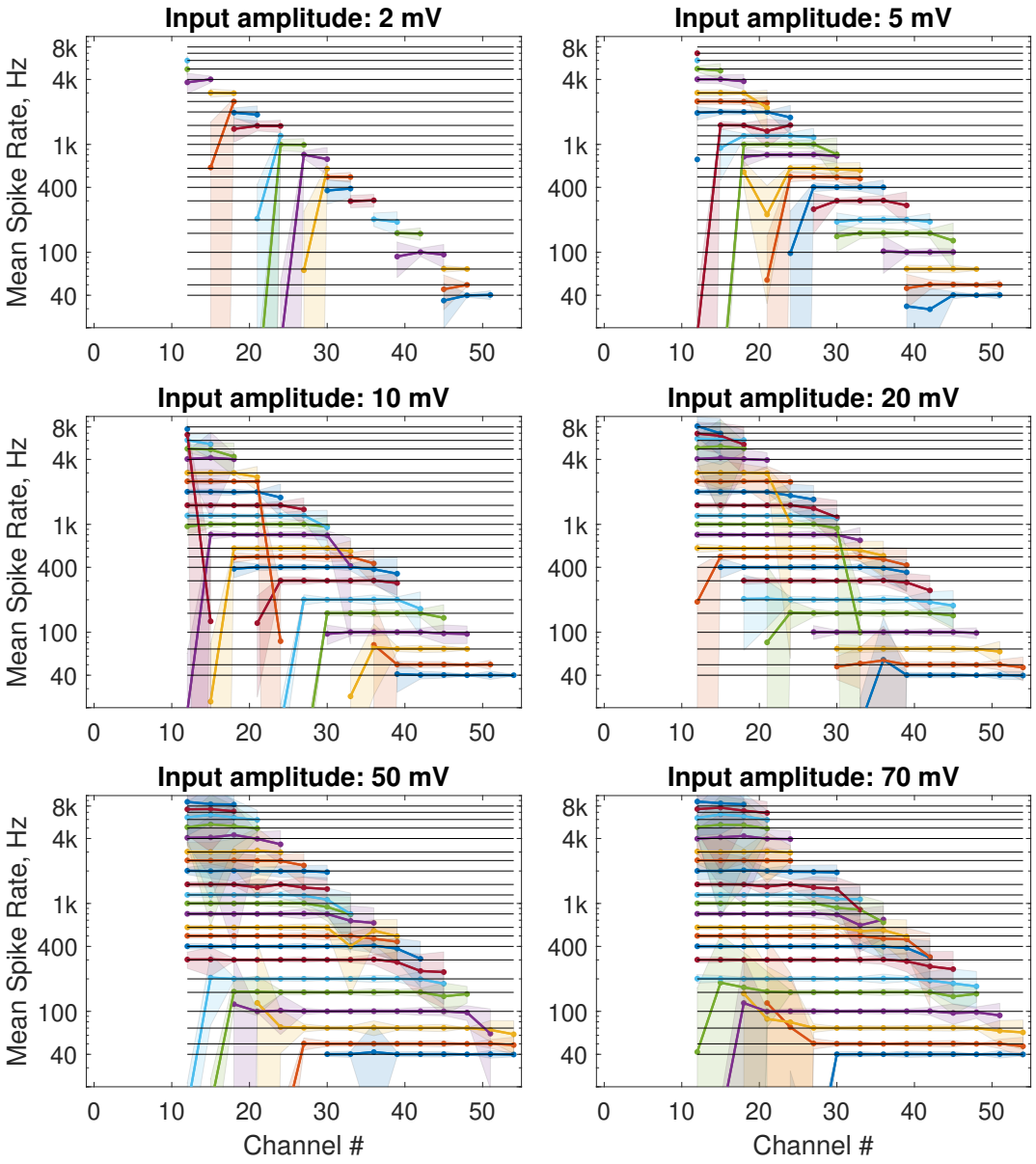

Figure 3.22: Spike frequency response obtained from filtered ON spikes in the AGC case. These plots are analogous to the plots in Fig. 3.20 and the filtering is done the same way. The plots show that AGC helps to suppress the oscillation at the high frequency channels in response to high-amplitude low frequencies, mentioned at the Fig. 3.20 caption, however, there is an increased variance of the response at the low-frequency channels to a high-frequency signal, which is indicated by the increased shadowed region at the diagonal, compared to Fig. 3.20. This is due to the fact that spike rate is normalized not by actual signal frequency, but by the characteristic frequency of the channel, so, when even a low-amplitude high-frequency signal shows up at the low-frequency channel, the spike rate easily exceeds the threshold, and the channel gain is decreased to the level when the signal completely disappears. Then the AGC has to increase the gain again, and the process repeats, causing gain oscillations and increasing the variance of the spike rate.

3.4.4 AGC Transient Measurements to Speech

To demonstrate the change in the spike rates from the local AGC in response to dynamically changing input signal amplitudes for natural sounds like speech, we plotted the input signal and the analog output of the bandpass filter of one channel (#30, 642 Hz) in Fig. 3.23 (top) in response to a speech sample. The speech spectrogram and the gain-corrected cochleagram for 36 out of the 64 channels are plotted in the middle section. The channel spike count response and the gain change events in response to this signal are plotted in the bottom section of Fig. 3.23. The gain change events (red dots) indicate gain setting changes when the spike count in the time window has exceeded one of the 2 thresholds. The ordinate of the dot shows a new gain value for the selected channel. When the channel gain reaches the extreme values (0 or 32.5 dB), it no longer changes, thus one can see two gain change events at the same gain level at the beginning of the plot. The blue dots indicate the cochlea ON-spikes at the selected channel, their ordinate represents the current gain of the channel at the time when the event was generated.

This plot shows that at the beginning, the channel produces no spikes because there is no input signal, therefore the gain stays at the highest setting. When the speech sample starts, the output amplitude is initially high because of the high gain. Over a period of about 150 ms, the gain value decreases in steps until the spike count (yellow bars in Fig. 3.23) drops below two spikes per period. The spike count represents the number of cochlea spikes per period of channel #30's characteristic frequency averaged over a window of 16.6 ms. The width of the bars is equal to the length of the averaging time window. The gain setting stays constant at 3 dB during the period of $0.2 - 0.5$ s, when the spike rate fits within the desirable range, and then gradually increases close to the maximal value during the period of $0.56 - 0.68$ s, when no spikes are produced at this channel.

The spike responses of 8 other channels for the same speech sample are plotted in Fig. 3.24. The fitted characteristic frequency for each channel is given next to each plot. The fitting of the characteristic frequencies is described in Appendix A.2.7. The top plot shows the input speech sample. The next plot shows the spectrogram for 36 frequency bands. Channel #46 does not have any gain change events because the amplitude of the signal at this frequency is low. The gain settings of the next 2 channels decrease but do not reach the minimum value of 0 dB. The gain setting for channel #24 decreases all the way to 0 dB as expected because of the higher signal amplitude at this frequency. The gain setting increases again around 0.3 s corresponding to the decrease in the signal power at this frequency in the spectrogram. The remaining channels see a similar trend in the decrease of the gain setting at the beginning but they do not reach the minimum value because of the reduced signal power at their frequencies. These channels respond faster to the onsets and offsets of the speech sample because of the smaller averaging time window.

These measurements show that this event-based AGC algorithm helps to keep the input amplitude within the linear operating range of the amplifier.

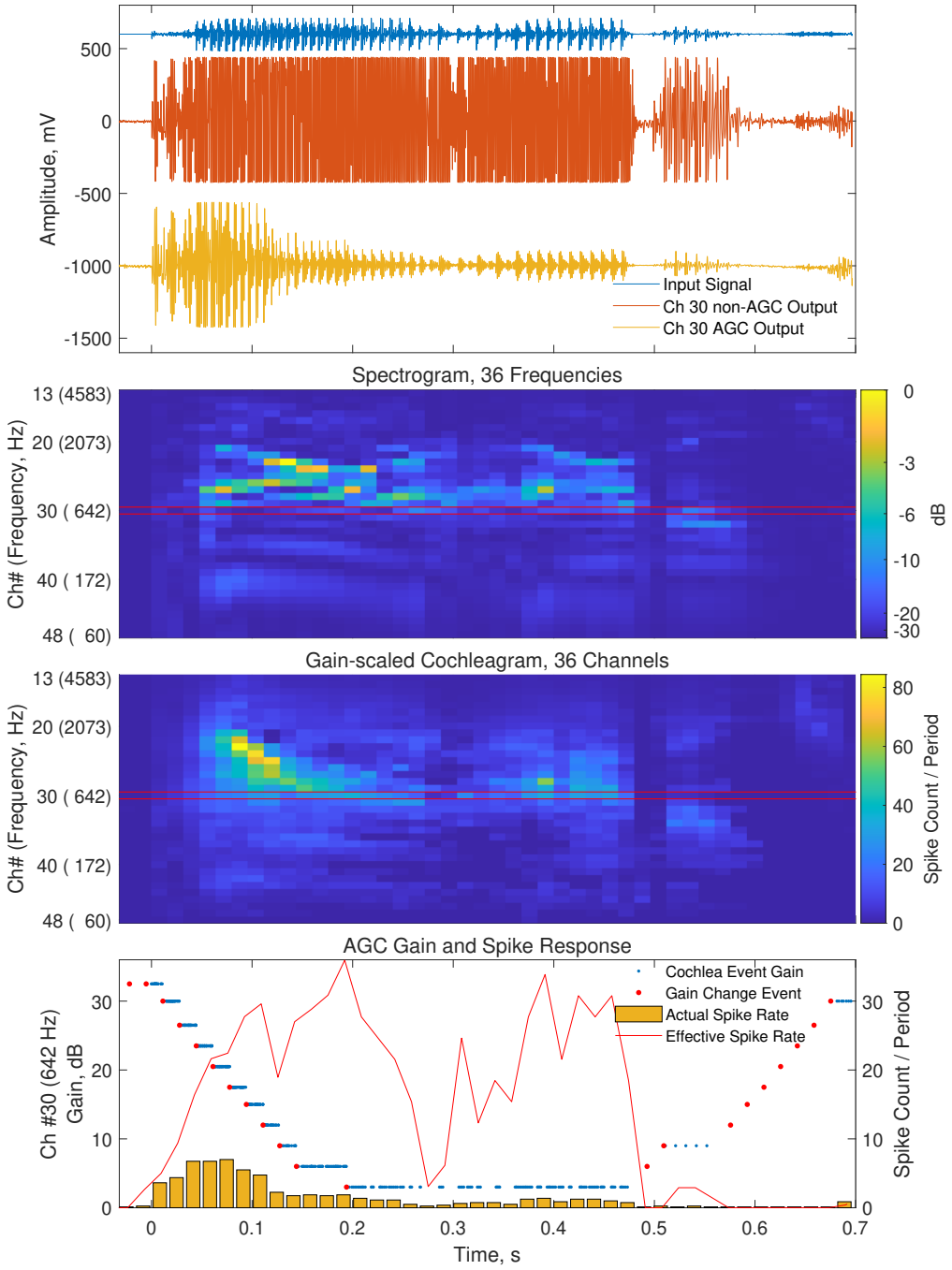

Figure 3.23: AGC response of channel #30 (642 Hz) to the speech sample "Power out" (male voice) from the TIMIT dataset. Top: Waveform envelope for input signal (blue) and channel output for non-AGC (red) and AGC (yellow) cases. The curves are plotted at an offset for visibility. Middle: Spectrogram for 36 frequency bands and gain-scaled cochleagram for 36 cochlea channels with the same best frequencies. Cochlea spike counts are scaled according to the current channel gain within each time bin. Bottom: AGC gain change events and spike responses. Red dots indicate the new gain value after the gain setting was updated. Blue dots represent ON-spikes, their y-axis value reflects the gain of the channel when the spike was generated. The yellow bars show an average number of spikes for one period of the channel's characteristic frequency. The width of the yellow bars is equal to the averaging window length. The thin red line shows the spike rate that would be needed in the non-AGC case to represent the channel's output signal at the maximal gain, assuming no signal clipping.

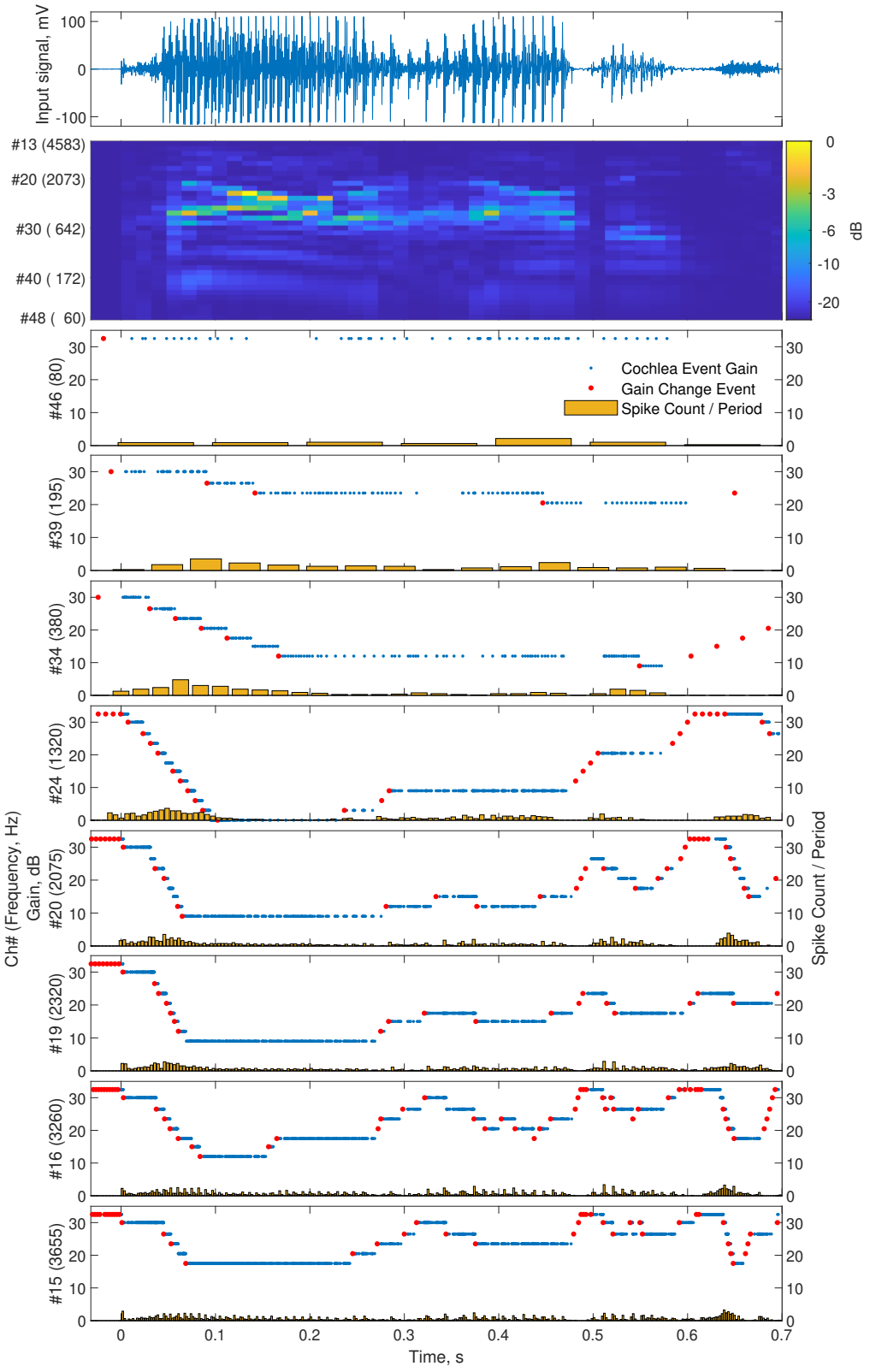

Figure 3.24: Gain change events and spike responses of 8 channels in response to the speech sample in Fig. 3.23. The same notation is used in this figure. The top plot shows the input speech sample. The next plot shows the spectrogram for 36 frequency bands. The gain for each channel changes in a different way following the signal amplitude at the corresponding frequency. The higher frequency channels (#15 – 20) respond faster to the onsets and offsets in the speech because of their shorter averaging window. The spike rates for all channels remain low.

3.4.5 Input Features and Classifier

We evaluate the features generated from the spike activity of the silicon cochlea output in a classification task of detecting speech versus noise. Different spike features have been been proposed for driving the subsequent classifiers. These features include constant time spike count, interspike interval histograms and constant count features (Anumula, Neil, Delbruck, *et al.*, 2018; Li, Delbrück, *et al.*, 2012).

In this work, we use inter-spike interval histogram and spike count features similar to the features described in Li, Delbrück, *et al.*, 2012. The features consist of the 80-bin histogram of inter-spike intervals, event counts of 36 channels, and 36 values representing the average gain of each channel, all computed within a 400 ms frame without overlap. Inter-spike intervals for all channels are computed separately and then combined into one histogram. Inter-spike intervals that are greater than 150 ms are excluded before the histogram is computed. For the "non-AGC" case, the 36 values representing the average gain of each channel are set to a constant value. The resulting 152-dimensional feature vectors are then used to train a binary Logistic Regression (LR) classifier.

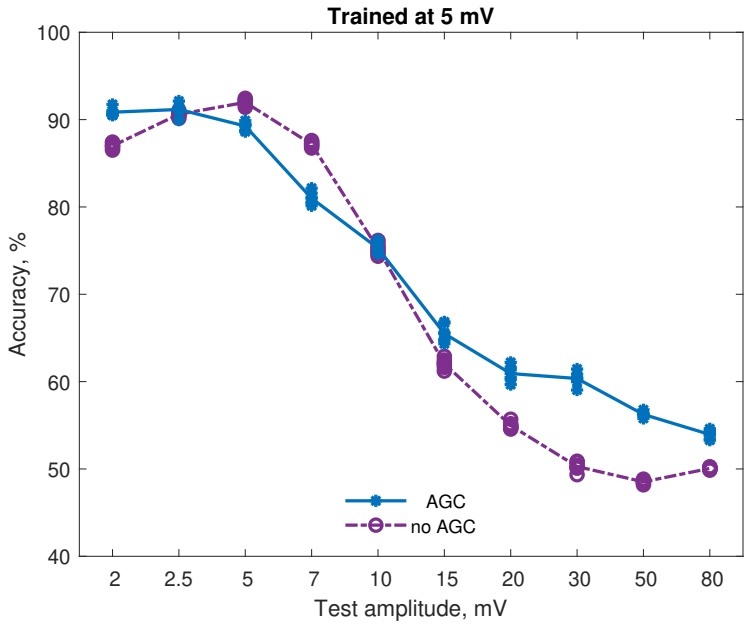

Figure 3.25: Speech vs noise accuracy of a classifier trained on recordings at 5 mV input and tested at other amplitudes.

3.4.6 Voice Activity Detection Classification Task

We compare the accuracy results for the LR classifiers trained on the spike responses of the cochlea operated in either the AGC or the non-AGC mode. Figures 3.25 to 3.29 show the speech versus noise accuracy of the classifiers trained on the features acquired at five different input amplitudes: $[5, 10, 15, 50, 80]$ mV respectively. The plots show that if the test data is acquired at the same signal amplitude as the train

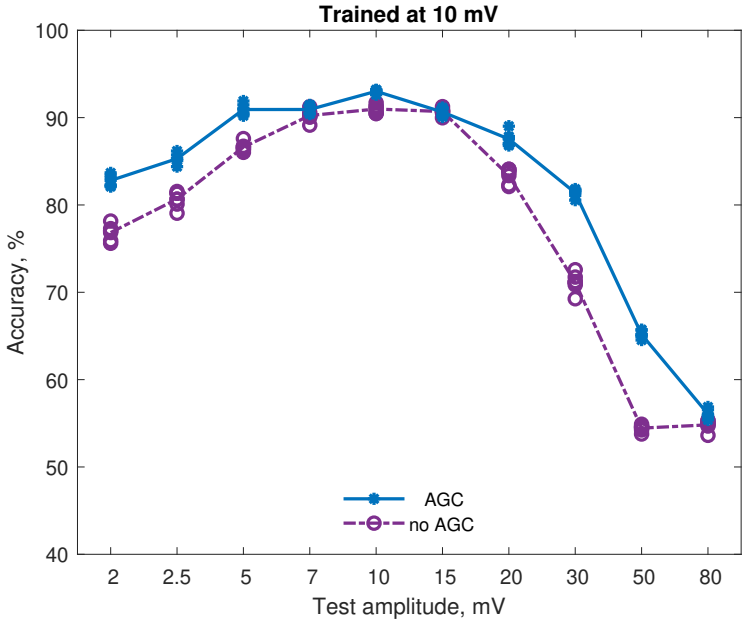

Figure 3.26: Speech vs Noise accuracy of a classifier trained on recordings at 10 mV input and tested at other amplitudes.

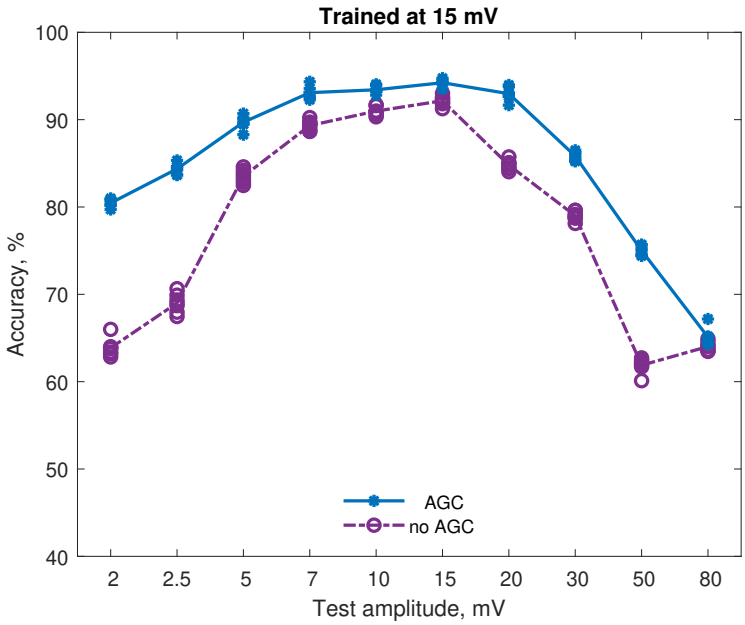

Figure 3.27: Speech vs Noise accuracy of a classifier trained on recordings at 15 mV input and tested at other amplitudes.

data, then the test accuracy is about the same for the AGC and the non-AGC cases. However, when the test data is acquired at higher or lower amplitude, the accuracy drop of the classifier for the non-AGC case is much higher than for the AGC case. The only exception is observed at Fig. 3.25, where the 5 mV classifier accuracy for the AGC case is lower than non-AGC accuracy, this can be caused by the transient processes when the AGC switches between some attenuation levels, as described in Sec. 3.2.2. The classifier trained for the AGC case achieves higher accuracy than one

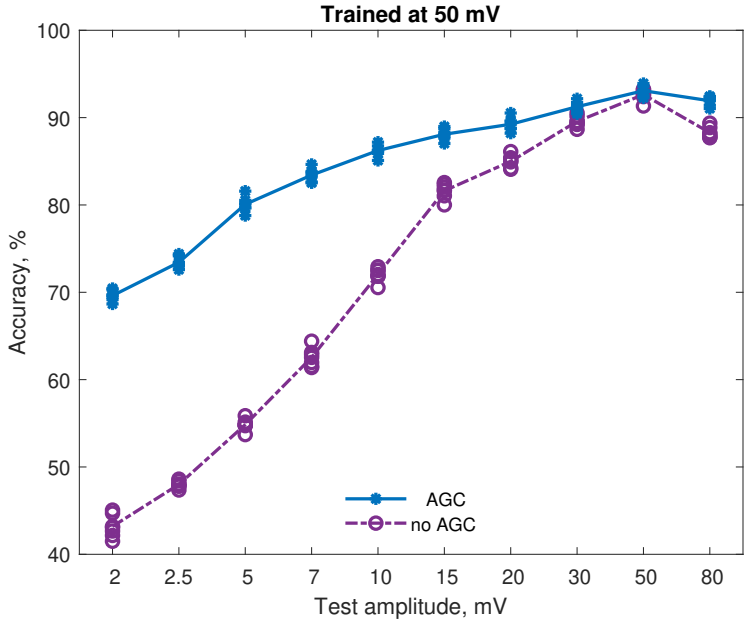

Figure 3.28: Speech vs Noise accuracy of a classifier trained on recordings at 50 mV input and tested at other amplitudes.

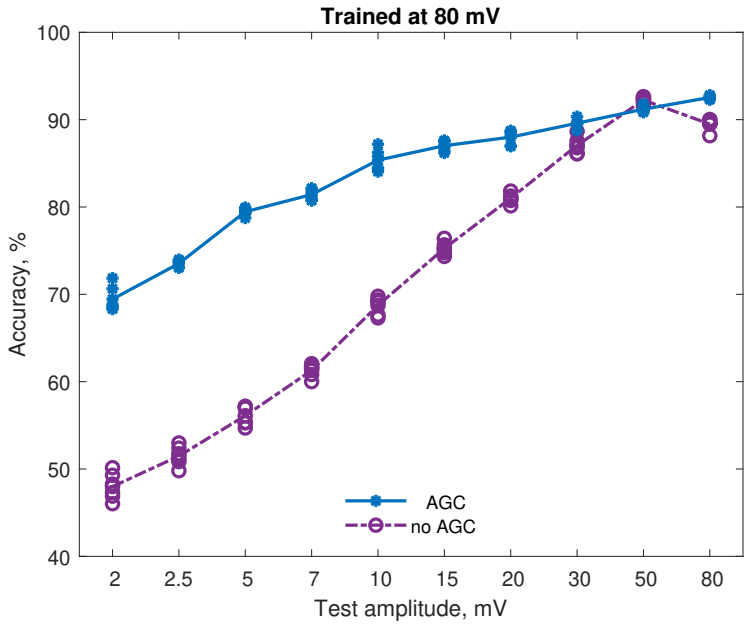

Figure 3.29: Speech vs Noise accuracy of a classifier trained on recordings at 80 mV input and tested at other amplitudes.

trained for the non-AGC case, when tested over all input amplitudes. A summary of the average accuracy improvement of the models trained at different input amplitudes and tested over the full range of amplitudes is shown in Fig. 3.31. The results show that running the cochlea in the AGC mode improves the accuracy of the classifier by about 10% when the test and train amplitudes differ by just 3 dB, and the accuracy improvement grows further up to 40% when the difference between train and test signal amplitudes increases.

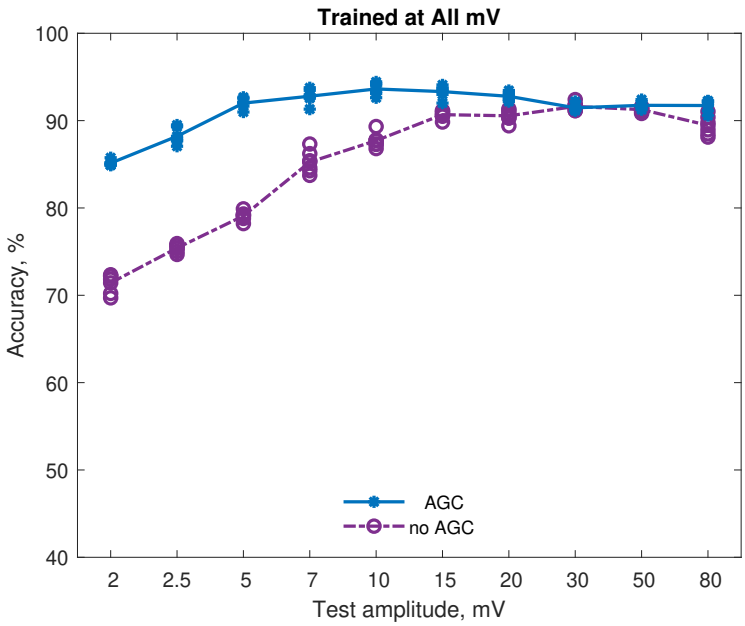

Figure 3.30: Speech vs Noise accuracy of a classifier trained on recordings at all amplitudes and tested at each amplitude separately.

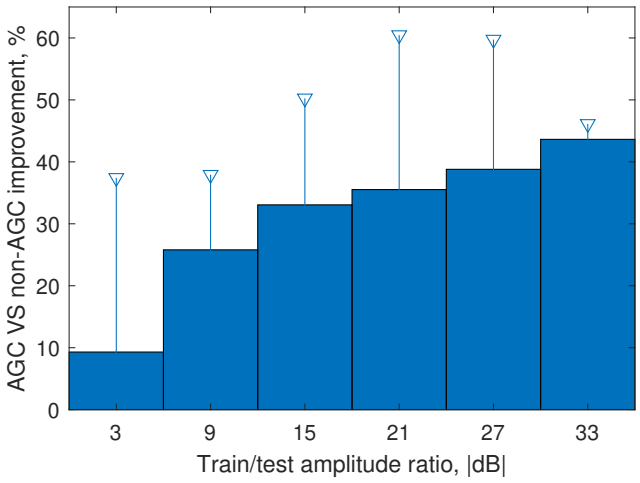

Figure 3.31: Average accuracy improvement of speech/noise classifiers trained at different input amplitudes for the AGC vs non-AGC case, when tested over a wide range of input amplitudes.

3.5 DISCUSSION

The event-driven AGC algorithm described in this chapter can help to reduce the event rate of a spiking cochlea by 4x over a wide range of input amplitudes and ensures that the transistor circuits still function as the small-signal bandpass filter function. It also therefore preserves more information which we demonstrated through a classification task of speech versus noise. The results show that a trained classifier shows higher accuracy performance on spike inputs from an AGC-controlled spiking cochlea when tested over a wide range of input amplitudes. We presume that AGC can help to preserve a shape of inter-spike intervals distribu-

tion of a channel in the presence of an interfering signal at the adjacent frequency bands. Initial studies on this topic were carried out and the results are described in Sec. A.2.8 of the Appendix. The preliminary study of how the shape of instantaneous frequencies histogram at one channel (channel 30, 480 Hz, 5 mV) changes in the presence of a 1 kHz interfering frequency at two signal-to-interference (SIR) levels: 0 and -6 dB is described in (Kiselev and Liu, 2021).

The benefits of using spike features from the AGC-controlled spiking cochlea can be further demonstrated on a simple speech recognition task like the continuous digit recognition system reported in (Gao, Braun, *et al.*, 2019). In this system, the spike count features are used as input to a delta Recurrent Neural Network on during training. The RNN was implemented on an FPGA system therefore it will be easy to include the gain information provided by the gain control mechanism into the features using the same FPGA (Gao, Braun, *et al.*, 2019; Gao, Rios-Navarro, *et al.*, 2020). The AGC digital circuits as implemented on the FPGA here can be added to an ASIC chip targeted at audio edge applications, e.g, the Voice Active Detection chip that combines a spiking cochlea filter bank together with a multilayer perceptron (Yang, Yeh, *et al.*, 2019). It can be adapted for other spiking audio front ends that generate spikes for a SNN hardware platform, e.g (Kiselev, Neil, *et al.*, 2016a; Tsai *et al.*, 2016).

3.5.1 Improvements of Local AGC Mechanism

With the current mechanism, it is not possible to use it with high-frequency channels because the gain update rate of these channels would be higher than is possible with the current DASLP design. This can be mitigated by increasing the averaging window length, but that would lead to increased attack and release times at low frequency channels. One possibility is to try using measured interspike intervals in the future.

We considered a possibility of using estimated signal frequencies for defining the averaging windows for the channels, instead of using the fixed length windows computed from the channel characteristic frequencies. However, each cochlea channel responds to a wide range of frequencies, especially when input signal amplitude is high (see Sections 3.4.2, 3.4.3) and measuring of actual signal frequency at a channel is complicated even in case of a single frequency signal. Even though it is possible to measure the signal frequency by filtering spikes as described in Sec. 3.4.3, it would give a subtle advantage for the AGC algorithm. For the low frequency channels it would be possible to decrease the attack and release times, when a high frequency interfering signal is present. However, using actual measured frequencies does not eliminate the need for using time window counters for each channel, because we have to stop the spike rate averaging at some point, when there are no spikes at all at the channel. For the high frequency channels these averaging time limits are much shorter than a potential interfering frequency period, so measuring that period would be useless for the high frequency channels.

Other future studies can be carried out to determine how an increased spike-rate averaging window and hence, increased attack and release times, affect the AGC performance. Increasing the time averaging window would help to improve

the AGC loop stability but increased attack and release times may deteriorate the advantage given by AGC. The attack time can be reduced by continuous comparison of the channel spike counts with the upper threshold and early interruption of the averaging when the threshold is exceeded. However this approach requires more FPGA resources and complicates routing. The proposed method renders the attack and release times to be close to each other. If a longer release time is required this can be accomplished by adding a low-threshold-breakout counter to each channel. That would increase the release time by the factor specified in the counter.

Another possibility for improving AGC stability might be adjusting the upper threshold for the spike rate, or making the threshold individual for each channel. It can be even varying in time, e.g. decreasing the threshold in an absence of signal would lead to a faster AGC response to the signal onset, and increasing the upper threshold in a presence of the signal would eliminate unnecessary gain changes during abrupt but short surges of the signal amplitude.

It is possible to implement the proposed AGC mechanism using two analog LIF neurons — one for the upper threshold, the other for lower one. Computation of the average spike rate over a time window can be represented by a membrane potential of the LIF neuron with an excitatory synapse. When the membrane potential reaches the upper threshold, the neuron elicits a spike, which indicates that the channel gain has to be decreased. For the lower threshold a neuron with intrinsic spiking activity and inhibitory synapse is needed. When this neuron receives enough spikes from the channel output neuron, its intrinsic spiking activity is suppressed, however, when it does not receive any spikes within a defined time period, it starts spiking, indicating that the gain has to be increased. The time constants of these neurons should be selected based on the channel characteristic frequency.

4 MULTI-MICROPHONE PLATFORM FOR AD-HOC NETWORK

This chapter describes a distributed WASN platform called WHISPER that is capable of synchronous multichannel audio sampling at different spatial locations with a sampling clock phase difference less than 200 ns and a jitter below 9 ns rms. The platform comprises up to four data acquisition modules with onboard computing capabilities, that can form an ad-hoc Wi-Fi network allowing an additional processing module such as a laptop or a smartphone to be connected if needed. Each acquisition module holds four digital Inter-IC Sound (I^2S) microphones with a total of 16 microphones for the entire system. Wireless synchronization of the sampling clock on each platform is implemented using a separate wireless module operating in the $902 - 928$ MHz ISM band. Usage of this system is demonstrated in a real-time application involving spatial sound filtering through a beamforming algorithm.

A large part of this chapter comes from a paper titled "WHISPER: Wirelessly synchronized distributed audio sensor platform" presented at IEEE SenseApp 2017 (Kiselev, Ceolini, *et al.*, 2017a) and a co-authored paper titled "Evaluating multi-channel multi-device speech separation algorithms in the wild: a hardware-software solution" which is published in the IEEE Transactions on Audio, Speech, and Language Processing journal (Ceolini, Kiselev, *et al.*, 2020). The author has only used the text related to the work contributed by the author in the papers.

4.1 BACKGROUND

The multi-channel acoustic sensor platform WHISPER was developed within a multi-partner European project called "Cognitive Control of a Hearing Aid" or COCOHA (www.cocoha.org). This project aims to help the hearing impaired in difficult auditory scenes by using attentional cues extracted from EEG signals to steer attended auditory sources by applying signal processing methods that are based on microphone arrays. Besides using the microphones on the hearing aid of a listener, an ad-hoc multi-microphone array is used for general acoustic scene analysis as presented in Fig. 4.1.

At the start of project, there were not known ad-hoc multi-microphone arrays that allow microphones to be placed randomly around a space. In many cases, the microphones were placed in a fixed geometrical setting so that beamforming algorithms can be applied easily because the geometry is known. The WHISPER multi-channel platform assumes no geometrical constraints between the microphones, and that sampled microphone signals from one module (WHISPER-M4) can be transmitted wirelessly to a central processing unit or to each other. The platform development is targeted at real-time low-latency spatial filtering with the aim for the use to speech enhancement and to supplement hearing aid devices.

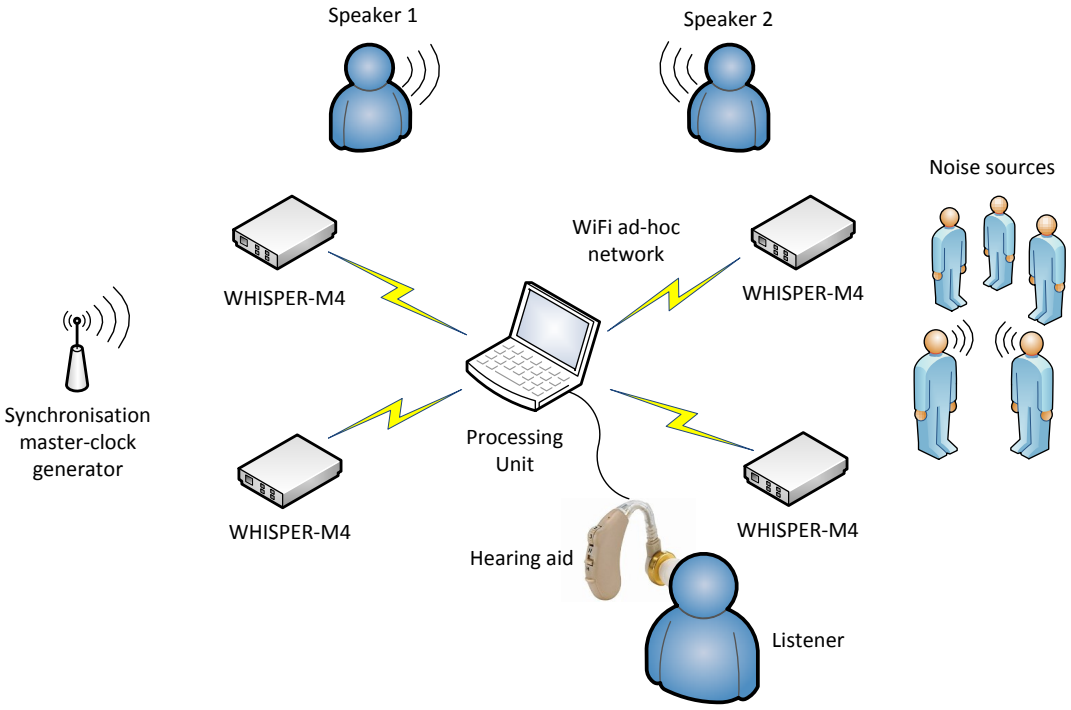

Figure 4.1: WHISPER platform usage scenario

4.1.1 Challenges of Platform Development

We made an early decision to design a platform that is composed of four spatially separated modules which are placed within a 3×3 m^2 space in a room, and the modules would be connected through a wireless network. This design imposes the following challenges involving sampling clock synchronization, synchronization of packets received wirelessly from different modules, and wireless communication latency.

Sampling clock synchronization across modules is an issue when processing audio streams from nodes with different oscillators, because the time alignment between the streams is unknown and even drifts with time. Synchronous microphone sampling on different modules is important for algorithms such as blind source separation (BSS), where previous work showed significant drops in performance with non-synchronized microphones (Lienhart *et al.*, 2003). Synchronization is also crucial for source localization and segregation based on sensor array geometry.

The work in (Girod *et al.*, 2006) reported a solution for time synchronization of 10 μs precision across multiple nodes, but the platform does not provide sampling clock synchronization. The work in (Schörkhuber *et al.*, 2014) uses a wireless sampling clock synchronization method based on a sub-GHz ISM transmitter and a subsequent Frequency Locked Loop (FLL) similar to the approach proposed in this work, however their method provides only 20 μs precision in the sampling phase synchronization. The prototype of a commercial acoustic sensor system announced in (*3D Audiosense* 2014) is claimed to have a sampling clock synchronization across the nodes, but without specifying any numbers. Another commercial product (*Lockit* 2019) uses a proprietary network technology — ACN, developed by Ambient

Recording GmbH, and provides phase synchronization accuracy of 8 μs. Also, both mentioned commercial solutions do not specify the latency of audio acquisition.

Due to the low-latency requirements of the intended platform, we cannot apply the blind synchronization technique proposed in (Miyabe *et al.*, 2015). It is also impossible to use either Network Time Protocol (NTP) or Global Positioning System (GPS) signals for the sampling clock synchronization because NTP does not provide sufficient accuracy and GPS is not available inside buildings (Sazonov *et al.*, 2010).

Latency of wireless transmission is also a critical factor. Ideally, transmitted signals should arrive no later than propagated acoustic signals to avoid a time difference between the transmitted sound and the direct sound. This is important for people who have mild hearing loss at lower frequencies and use non-occluding hearing aids (Winkler *et al.*, 2016). If the transmitted signal arrives with a delay, acoustically transmitted noise cannot be perfectly cancelled. Constraints of wireless technology make this ideal of faster-than-sound transmission difficult to achieve, and therefore one may need to settle for the lesser goal of minimizing the perceptual mismatch between acoustic and visual cues. The estimation of a Point of Subjective Simultaneity (PSS) and a just noticeable difference (JND) for audiovisual stimuli varies in a wide range across different studies and types of stimuli. The JND ranges from 20 ms in (Hirsh *et al.*, 1961) up to around 100 ms in more recent studies (Ipser *et al.*, 2017; Kawase *et al.*, 2016; Van Wassenhove *et al.*, 2007; Vroomen *et al.*, 2010). Overall, results suggest that JND time for auditory stimuli delays is higher for audiovisual stimuli of higher complexity, such as lip movements, words and phrases, compared to flashes and beeps. So, in this work we are targeting $20 - 100$ ms latency range for the whole data acquisition and processing pipeline.

While developing the WHISPER platform we investigated possible solutions for three following challenges:

- Synchronization of sampling clock frequency and phase across different modules of the platform. Our solution is presented in Section 4.3, in particular, the final solution is described in subsection 4.3.3.

- Synchronization of wirelessly transmitted data packets at the processing module. We investigated the use of Wi-Fi for wireless transmission of data between modules. Because of its asynchronous transmission manner and unpredictable data delivery latency, a packet level synchronization solution is needed. We handle this problem by broadcasting the "Start of Acquisition" signal to all the modules via the synchronization signal generator as described in Sections 4.3.2-4.3.2.3.

- Minimization of transmission latency of audio data packets between the modules of the platform. Measurements of the transmission latency using Wi-Fi (presented in Section 4.2.4) show that another solution for wireless data transmission is needed. The possible solution is discussed at the end of Section 4.2.4.

4.2 WHISPER PLATFORM

The WHISPER platform consists of up to four identical WHISPER-M4 modules and a synchronization signal transmitter. Each of the WHISPER-M4 modules can accommodate up to 4 digital microphones. All the modules perform synchronous sampling of the audio signal and transmit the data streams wirelessly to a processing device.

The distinct features of the WHISPER platform are: (i) its wireless modular design; (ii) synchronous sampling of multiple spatially distributed microphones; (iii) local processing capabilities (FPGA+software); (iv) potentially low latency of data transmission; (v) relatively low power consumption; and (vi) extension possibilities.

4.2.1 WHISPER Hardware

Two versions of the WHISPER-M4 modules are developed and used within this work. The prototype version of the WHISPER hardware is built solely out of COTS components, although, with little soldering required (see Appendix A.3 Fig. A.23). It supports 12-bit SPI microphones PMODMIC3[1] from Digilent and sampling frequencies 8 and 16 kHz. The final version supports 24-bit I^2S microphones from InvenSense[2,3], which are not available in the Peripheral Module (PMOD) format, so, a custom Printed Circuit Board (PCB) for holding the microphones is developed. The final version can also support the old SPI microphones if needed. Both versions were used for experiments described it the Section 4.4. First we describe the final version of the hardware and later in Section 4.2.2 we specify in which aspects the prototype version is different.

A block diagram of the WHISPER-M4 module is depicted in Fig. 4.3 with a picture shown in Fig. 4.2.

The two basic building blocks are a CPU-based computing unit, the Raspberry Pi 3 Model B+[4], and an FPGA board, namely the LOGI-PI-2[5] board from ValentF(x). The most recent version of the Raspberry Pi (Raspberry Pi 4 Model B) can be used as a drop-in replacement if more compute power is needed. The Raspberry Pi was chosen because of the existence of LOGI-PI-2, an FPGA board which has been developed specifically for interfacing to the Raspberry Pi. Both computing platforms are necessary, because they provide a different and complementary set of interfaces.

The FPGA is used to implement the synchronization algorithm and to extend Input-Output (IO) capabilities of the Raspberry Pi board. It allows synchronous sampling of an array of digital microphones, as well as other sensors with different interfaces, e.g. accelerometers (Zohourian *et al.*, 2016). The FPGA can also be used to do some signal pre-processing, such as digital filtering, downsampling, digital

1 https://reference.digilentinc.com/_media/reference/pmod/pmodmic3/pmodmic3_rm.pdf
2 https://www.invensense.com/wp-content/uploads/2015/02/ICS-43432-data-sheet-v1.3.pdf
3 https://www.invensense.com/wp-content/uploads/2016/02/DS-000069-ICS-43434-v1.2.pdf
4 https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus
5 http://valentfx.com/logi-pi

Figure 4.2: WHISPER-M4 module for COCOHA system with attached custom digital I^2S microphones powered from a USB powerbank.

AGC, and data manipulation, such as compression, scrambling and error-correction codes.

The Raspberry Pi runs an Operating System (OS) and offers a way of embedding into the platform floating point computations, which would be harder to implement on the FPGA. It also provides a set of network interfaces including Wi-Fi, Bluetooth and Ethernet.

The remaining hardware components are custom-made digital microphones and the Radio Frequency (RF) module. The digital microphones are based on 24-bit MEMS microphone from InvenSense — ICS-43432[6] (Fig. 4.4d), but only 16 bits [19:4] are used in this work. The microphone ICS-43434[7] is also supported. Both microphone models have high SNR of 65 dBA and sensitivity −26 dBFS. The custom microphone schematic is given in Appendix A.3 Fig.A.22). Custom-made PMOD-to-RJ45 adapter (Fig. 4.4c) allows to connect the microphones to the WHISPER-M4 module with standard Ethernet cables. The RF module is the MRF89XAM9A[8] transceiver from Microchip Technology Inc. used for implementation of the synchronization algorithm which is described in Section 4.3.

To implement wireless synchronization of data sampling across different modules of the platform, as described in challenge 1) in Section 4.1.1, we use the MRF89XAM9A transceiver module operating in ISM 902 – 928 MHz frequency band and providing up to 200 kbit/s data rate utilizing frequency-shift keying (FSK)

6 https://www.invensense.com/wp-content/uploads/2015/02/ICS-43432-data-sheet-v1.3.pdf
7 https://www.invensense.com/wp-content/uploads/2016/02/DS-000069-ICS-43434-v1.2.pdf
8 https://ww1.microchip.com/downloads/en/DeviceDoc/75017B.pdf

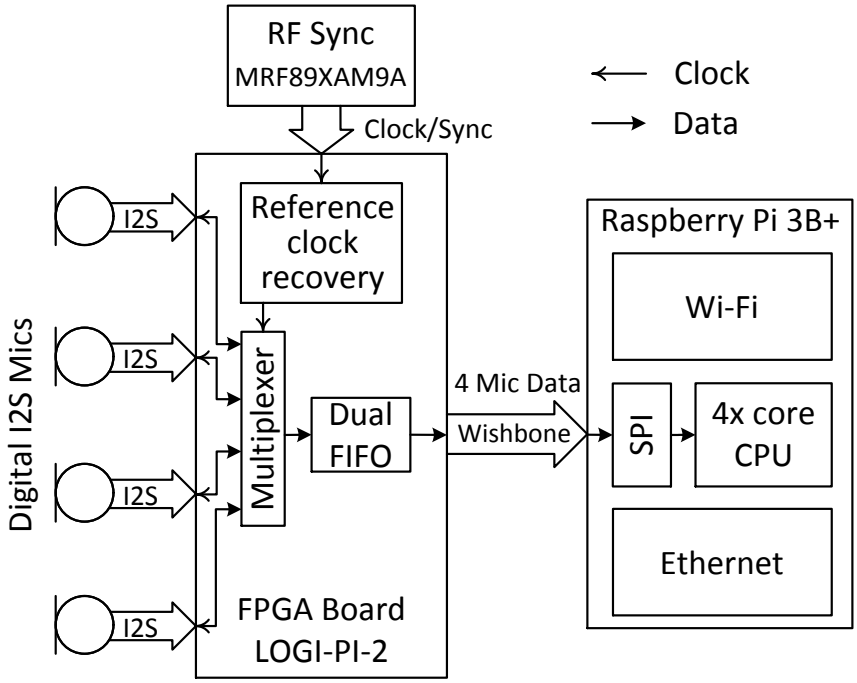

Figure 4.3: Block diagram of the WHISPER-M4 module.

modulation. The schematic of the module is given in Appendix A, Fig. A.24. This module, when operated as a transmitter, can transmit a continuous stream of data via 2-wire interface consisting of DCLK (clock) and DATA signals. When operated as a receiver, it allows access to the clock recovered from the transmitted data. This recovered clock plays a crucial role in our synchronization algorithm. Most of the other data communication modules only provide packetized data therefore precise timing information is impossible to recover. Although the MRF89XA chip used in the transceiver has both DATA and DCLK signals necessary for the continuous data transmission mode, the MRF89XAM9A module does not provide a dedicated pad for accessing the DATA signal, however this signal is routed to the R2 resistor, so it can be accessed by soldering a wire to the resistor pad as shown in Appendix A.3 Fig. A.23.

The synchronization algorithm implemented using this transceiver is described in Section 4.3. The transceiver can also be used for unidirectional low-bandwidth data broadcasting from the synchronization master to all other modules.

There are three options for wireless connectivity: 2.4/5 GHz Wi-Fi (IEEE 802.11n/ac), Bluetooth 4.2/BLE and Ultra-Wideband (UWB) 3.5 – 6.5 GHz transceiver DWM1000[9] (IEEE 802.15.4). Wi-Fi and Bluetooth are implemented using Cypress CYW43455 chip integrated in the Raspberry Pi 3 model B+. DWM1000 (shown in Appendix A.3 Fig. A.26) is an external module from DecaWave connected to FPGA via Serial Peripheral Interface (SPI) bus. It provides the data transmission rate up to 6.8 Mbit/s, which is enough to collect the data from 16 16-bit microphones sampled at 24 kHz. Although we investigated a possibility of using the DWM1000 module for

9 https://www.decawave.com/sites/default/files/resources/dwm1000-datasheet-v1.3.pdf

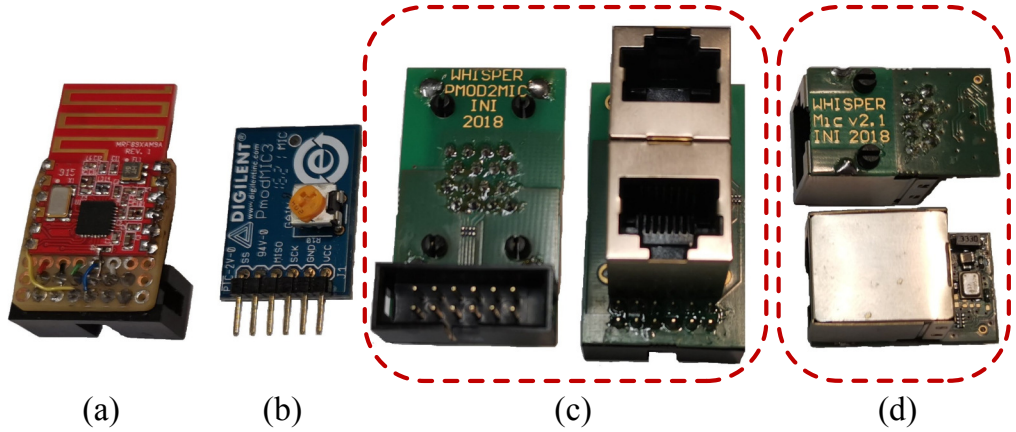

Figure 4.4: Peripheral parts of the WHISPER module: a) Synchronizer module; b) PMODMIC3 SPI microphone; c) custom PMOD to RJ45 adapter; d) custom I^2S microphone.

low-latency data transmission, it was not incorporated into the final version of the hardware, because the COCOHA project was ended.

4.2.2 WHISPER Prototype Hardware

Using the COTS components for building a prototype of the WHISPER platform allowed rapid prototyping of algorithms without need for development of any custom hardware. The prototype is based on the earlier version of Raspberry Pi — Raspberry Pi 3 Model B[10], which has only single-band (2.4 GHz, IEEE 802.11b/g/n) Wi-Fi chip — Broadcom BCM43438. We used PMODMIC3 digital microphones (Fig. 4.4b) from Digilent, which hold analog MEMS microphones (Knowles Acoustics SPA2410LR5H-B) and 12-bit ADC from Texas Instruments[11]. The microphones have SNR = 63 dB and sensitivity −38 dB @ 94 dB Sound Pressure Level (SPL).

4.2.3 FPGA Logic and Software

The FPGA is connected to the Raspberry Pi via a high-speed SPI bus. At software level, the Wishbone interface is used, which is based on the standard Wishbone libraries provided with the LOGI-PI-2 board. The maximal required data throughput of this interface for four microphones sampled at 48 kHz and 16 bit resolution is 3.072 Mbit/s. In order to minimize the communication overhead and to achieve such a data rate, we implemented a dual First In, First Out (FIFO) buffer in the FPGA and used the burst mode of the Wishbone interface (see Appendix A.3 Fig. A.19 for the address space). Using the burst mode helps to avoid sending an address every time we need to read a data word, but introduces some latency due to the time needed for filling the buffer.

10 https://www.raspberrypi.org/products/raspberry-pi-3-model-b
11 https://www.ti.com/lit/ds/symlink/adcs7476.pdf

Since we target low-latency applications, we have to use rather small data buffers to minimize the latency of the data acquisition. However, it is difficult to achieve hard real-time requirements for data readout with a small buffer size even in a Linux-based OS. In order to reduce the latency introduced by the OS task scheduler we used a boot option "isolcpus = 3" which excludes the CPU core #3 from the set of cores available to the scheduler. Then we set the task that communicates to hardware specifically to the core #3 with a command "taskset -c 3 ./hardware_task". We found that we need at least 8 ms to reliably read the data from a program running in user-mode on an isolated CPU core. The buffer size for four microphones in this case equals to $0.008 \cdot 4 \cdot F_s$ 16-bit words, where F_s is sampling frequency. We round this value to the next bigger power of 2, so we use 256-word FIFO for 8 kHz sampling rate and 1024-word FIFO for 24 kHz.

The FPGA has two such FIFO buffers. While the FPGA is reading out the data samples from the four microphones and storing them into one buffer, the other buffer is available for reading out from the CPU side. If the CPU has not managed to finish reading out the data within the specified time window, the whole next data packet is dropped to ensure data consistency and prevent overwriting of a partially read buffer with new data. We use polling of the status flag “Empty” in the Wishbone address space (Appendix A.3 Fig. A.19, last row) to determine whether a new data portion is available. Once the Empty flag is cleared, the CPU can issue a burst read of the whole buffer via the Wishbone interface.

The Status Register bits [15 : 4] represent 12-bit packet counter, and bits [3 : 0] are the [Empty & Stop & Stalled & Overflow] flags. When the Overflow flag is set, it means that one or more packets were dropped after the current packet. The exact number of the dropped packets can be calculated by comparison of the packet counters of the current and the following packets (see Sec. 4.3.2.3).

4.2.4 Communication and Network

We investigated two ways of connecting the WHISPER–M4 modules as part of a wireless network using: 1) the 2.4 GHz Wi-Fi (IEEE 802.11n) standard in ad-hoc mode or 2) the DWM1000 transceiver with TDM media access. We excluded the investigation of Bluetooth even though it was available on the Raspberry Pi. In both cases, there are two topologies for the data transmission graph depending on the algorithm used for data processing — a star topology for the centralized algorithm and a full mesh for the distributed processing. When using Wi-Fi, the data transmission graph topology is defined by software.

Although Wi-Fi 802.11n has a high data throughput, the latency of the data transmission is unpredictable and varies a lot with the network load. Specifically, an environment with a lot of access points using the same frequency bands would cause a lot of conflicts thus increasing transmission delays. For real-time audio processing applications, the whole data processing pipeline latency should not exceed 20 to 50 ms (Keetels *et al.*, 2012), therefore the data transmission latency should be in the range of 10 to 20 ms.

We measured latencies of wireless data transmission over Wi-Fi in realistic scenario with required data rates and packet sizes (Fig. 4.5) using a star topology

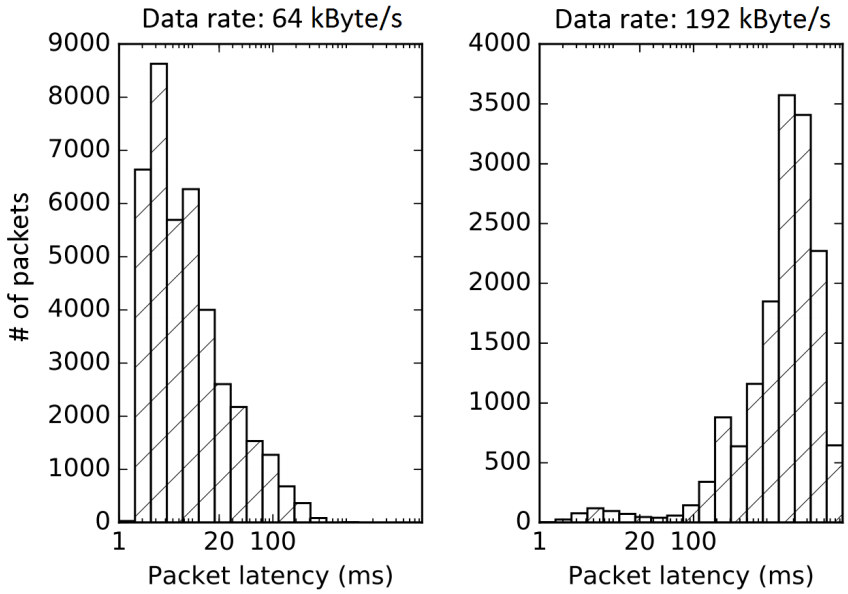

Figure 4.5: Measurement of Wi-Fi data transmission latency when four modules are transmitting simultaneously. The latencies are measured for the data rate needed for four 8 kHz microphones (left) and four 24 kHz microphones (right)

where data are sent from four WHISPER-M4 modules to a PC using User Datagram Protocol (UDP) protocol. We found that in most of the cases the latencies are unacceptably high, reaching hundreds of milliseconds in some cases. We presume that this happens because of data collisions since all boards may transmit data at the same time. This problem cannot be solved reliably by using 2.4 GHz Wi-Fi to our knowledge. Even though the Wi-Fi data transmission order can be controlled using custom firmware, there are always many other Wi-Fi enabled devices around that operate independently and would interfere in busy environments.

To meet the low-latency requirement, we added the DWM1000 wireless transceiver from DecaWave (IEEE 802.15.4 compliant). This transceiver provides direct control over transmission time allowing implementation of time-division multiplexing (TDM) media access therefore addressing challenges 2) and 3) in Section II. Since WHISPER-M4 modules are synchronized with sub-microsecond precision, we can implement a "circular" data transmission scheme, where each module has its dedicated time slot for data transmission and all other modules can receive data during this time. This scheme naturally implements the full mesh topology and is ideal for distributed data processing, although centralized processing also can be implemented. In the latter case, one additional WHISPER-M4 module with wired Ethernet connection to the processing unit is required to enable receiving data from four acquisition modules.

The data transmission latency measurements for the DWM1000 module are shown in Table 4.1. When all four WHISPER-M4 modules are used, these latency numbers increase by a factor of 4, but even in this case, these numbers are comparable with the acquisition time for this amount of data. Even though we evaluated the data transmission with the DWM1000 module, the results presented in the remainder of the paper are obtained by using Wi-Fi 802.11n.

Table 4.1: Measurements for DWM1000 latency

Packet size (bytes)	Packet length (μs)	Latency (μs)	Overhead (%)
128	150.5	318	113
256	301	542	80
512	602	828	37.5
1021	1201	1438	19.7

4.3 PLATFORM SYNCHRONIZATION

4.3.1 Background of Synchronization Algorithms

The challenge to putting a WASN in operation in real-time so that Multi-channel Audio Source Separation (MASS) algorithms can be deployed is not trivial.

Real-time online synchronization of the clocks on the local modules especially for ad-hoc arrays in unconstrained spaces, is still an active topic of investigation. A short review of various synchronization methods is given in (Schmalenstroeer *et al.*, 2015). Highly reliable synchronization solutions that operate across modules in an unconstrained physical space have been proposed in (Afifi *et al.*, 2018; Schmalenstroeer *et al.*, 2015) but many of these algorithms rely on offline estimations.

Many reported methods use a 2-way message passing algorithm that is needed because they estimate the propagation delays between modules in the network in order to synchronize the clocks. Our synchronization algorithm uses only unidirectional message passing where the reference clock signal is generated by a master clock generator. This clock signal is broadcast wirelessly to all the data acquisition modules. The receiving modules adjust the frequency and the phase of their sampling clock to match the received reference clock. In our target scenario of a medium-sized room ($5 - 10$ m on a side), we assume that the modules will be placed within an area of 3 m $\times$ 3 m, thus the difference in the radio wave propagation delay is only about 10 ns which is negligible compared to the audio sampling clock period of ≈ 40 μs (24 kHz).

4.3.2 Synchronization Algorithm of the WHISPER Prototype Hardware

In order to sample audio signals at different spatial locations we need to have a sampling clock signal with the same frequency and phase at these locations. Since we are targeting a wireless handheld spatially distributed application scenario, we cannot provide the sampling clock from one generator to all data acquisition modules over a wire. In order to accomplish the sampling clock synchronization across the spatially distributed acquisition modules we use a separate global clock generator, which transmits the DCLK clock signal to all the acquisition modules wirelessly (Fig. 4.6). This approach, however, requires precise clock recovery at the receiving sites.

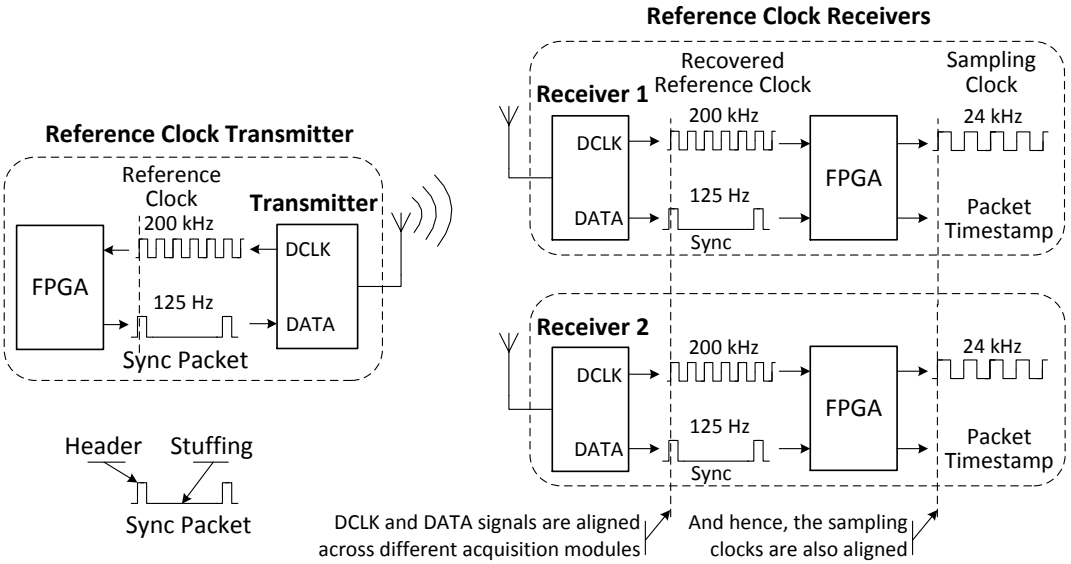

Figure 4.6: Wireless synchronization principle.

4.3.2.1 *Sampling Clock Synchronization*

The synchronization algorithm is based on a Phase-locked Loop (PLL) idea. The wireless transceiver MRF89XAM9A transmits the clock (DCLK) and DATA signals to all modules of the WHISPER platform (Fig. 4.6). The clock signal recovered from the data stream by the receiver is used to generate the sampling frequency. The data signal carries information for the packet-level synchronization. Since the transceiver supports a fixed number of data rates, it cannot transmit a clock signal of an arbitrary frequency. The highest possible data rate of 200 kbit/s was chosen because it gives the minimal jitter for the clock signal — about 150 ns.

In order to facilitate the clock recovery, the data stream has to have as many 0-to-1 and 1-to-0 transitions as possible. The data are structured into packets of 1600 bits that are transmitted continuously. The packets have 13-bit header and 7-bit packet counter, the rest of the packet is filled with alternating 0s and 1s. The packet headers in this case follow with 8 ms intervals and are used to recover from the clock misalignment condition, when one or several clock cycles are missing due to a noise in the radio channel.

In order to generate a phase-locked sampling frequency from the received clock signal, a digital PLL was implemented in the FPGA. This PLL generates a 48 kHz clock which can be divided by 2, 3 and 6 to get sampling rates of 24 kHz, 16 kHz and 8 kHz. If the sampling rates of 44.1 kHz and 22.05 kHz are needed, another PLL has to be used.

4.3.2.2 *Start of Acquisition Synchronization*

The sampling rate synchronization is not sufficient to align the received data from two modules. Due to random power-up times, different modules can start filling their data buffers at different times. This gives a constant shift between data samples coming from different modules at the receiving site. To avoid this misalignment, we

need to start data acquisition at all modules simultaneously. The "start of acquisition cycle" command is sent over the data channel of the synchronizer to all modules. When modules receive this command they reset their internal time counter, flush buffers and start data acquisition again. Each acquisition module counts the clock cycles coming from the synchronizer and embeds this time information into each data packet sent to the processing module so that data from all acquisition modules can be aligned at the receiving site.

4.3.2.3 Packet-Level Synchronization

By using the synchronization algorithm described in the previous section, the audio can be sampled from all the microphones synchronously. But when we use a wireless network to collect the data from different modules at the processing unit, the data packets might arrive scrambled and some packets could be missing. To address this problem, we developed an algorithm that merges the data coming from the multiple modules and that can deal with packets arriving from different modules in a scrambled order. In this algorithm, packets from all modules will be dropped, if one of them is too slow in transmitting the data.

Every module sends a packet which contains the packet ID $p_{id} \in [0, 4095]$ and a module ID $w_{id} \in [0, K-1]$ where K is the number of modules. The main structure we used to collect the data is a FIFO queue implemented with two dictionaries. We define the dictionary D^1 as having the following (key, value) pairs:

$$D^1 : (a, p_{id}) \tag{4.1}$$

where a is the arrival counter, which is updated every time a new p_{id} is received regardless of its w_{id}. We also define the dictionary D^2 as having the following (key, object) pair:

$$D^2 : (p_{id}, [(l_0, l_1, ..., l_{K-1}), n]) \tag{4.2}$$

where l_k is a byte array with the microphones samples from the module with index k, and n is the number of modules that already sent the packets with the same p_{id}, in other words it is the number of non-zero elements in the list $(l_0, l_1, ..., l_{K-1})$.

This solution with two dictionaries allows us to solve both the problem of having scrambled packets and also the problem of the rollover in the numbering of the packets. The latter is caused by the use of 12 bits to number the packets on the FPGA, thus inducing a rollover in the numbering after every 2^{12} packets. The following examples will clarify the procedure used to ensure that we can process all the received packets correctly.

4.3.3 Synchronization for the Final Hardware

4.3.3.1 Algorithm description for v2 hardware

The outline of the reference clock recovery algorithm on the receiving module is described next: The first step is to recover the clock from the two signals (RFclock, RFdata) transmitted by the RF module. In order to estimate the phase and frequency

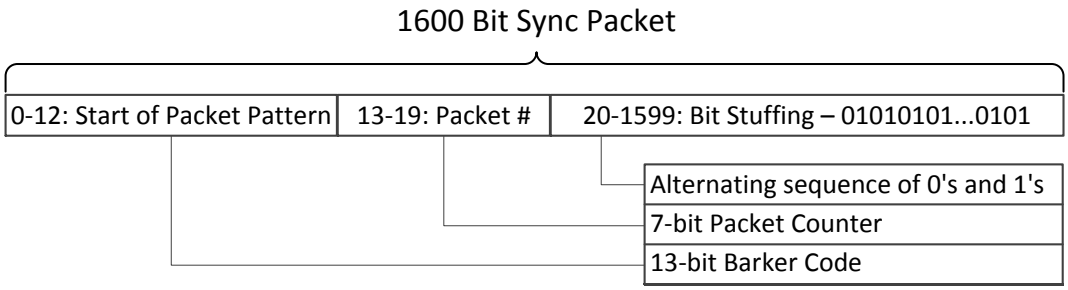

Figure 4.7: The synchronization packet structure.

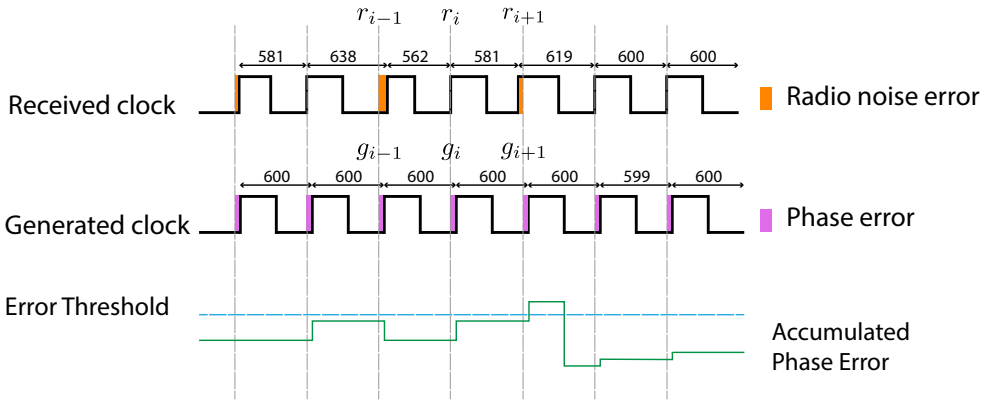

Figure 4.8: Signals from the synchronization algorithm implemented on WHISPER. The gray vertical lines delineate the period of the reference clock. The numbers above the waveforms depict the individual cycle lengths measured by the number of FPGA clock cycles.

of the reference clock, the RFclock is oversampled by the 120 MHz clock on the FPGA (Received clock in Figure 4.8). The recovered clock has the correct frequency and phase on average but the phase jitter of this clock is too high for our needs (see Figure 4.9 panel (a).

In an earlier version of our synchronization algorithm (Kiselev, Ceolini, *et al.*, 2017a), the RFclock is used to generate the audio sampling clock directly. The new algorithm described here introduces an additional step which first generates a new clock (Generated clock in Figure 4.8) that is phase-locked with RFclock but has much less jitter. This clock is then used to generate the audio sampling clock. Because of the reduced jitter, the noise in the sampled audio is decreased with respect to (Kiselev, Ceolini, *et al.*, 2017a).

The synchronization algorithm needs to guarantee that there is no difference in the frequency and little phase difference between the sampling clocks on different modules. Having only a small phase difference between sampling clocks is essential for algorithms that exploit knowledge of the relative positions of audio sources and microphones. These algorithms require implicit spatial location estimation with precision in the order of one degree. In order to fulfill this requirement, we aim to achieve with our synchronization algorithm, an inter-module clock phase

synchronization precision of less than 1 μs for a sampling frequency of 24 kHz. The synchronization algorithm addresses two problems: The first is to maintain a phase shift of less than one sampling clock cycle between clocks on different modules. The second is to have low clock jitter across time on each module.

To show how synchronization is achieved, we first define r_i and g_i to be the absolute times of the rising edges of the received and generated clocks respectively on the ith reference clock cycle (see Figure 4.8 for an example) and further assume that the frequency difference of these clocks is small.

By applying a quadratic loss function to the phase error of the generated clock on the ith clock cycle, i.e, $E_i = (r_i - g_i)^2$, the accumulated phase error over the most recent N clock cycles is described by

$$\mathcal{L}_i = \sum_{j=0}^{N-1} E_{i-j} = \sum_{j=0}^{N-1} (r_{i-j} - g_{i-j})^2. \tag{4.3}$$

We then estimate the phase shift, ϵ, needed for the generated clock so that the accumulated phase error $\mathcal{L}$ is minimized:

$$\mathcal{L}_i(\epsilon) = \sum_{j=0}^{N-1} (r_{i-j} - (g_{i-j} + \epsilon))^2 \tag{4.4}$$

$$\hat{\epsilon}_i = argmin_{\epsilon}(\mathcal{L}_i(\epsilon)) \tag{4.5}$$

A simple derivation shows that

$$\hat{\epsilon}_i = \sum_{j=0}^{N-1} (r_{i-j} - g_{i-j})/N \tag{4.6}$$

This phase shift is simply a moving average of the phase difference between received and generated clocks, which corresponds to a Finite Impulse Response (FIR) filter with the memory length N. In order to reduce the memory usage of the filter on an FPGA, we used an Infinite Impulse Response (IIR) filter that has similar characteristics but requires just one register. Simulations show that qualitatively there is no difference in the synchronization outcome. Thus, instead of computing moving average, we compute an exponential moving average of the phase difference over time:

$$\tilde{\epsilon_i} = (r_i - g_i)k_1 + \epsilon_{i-1}(1 - k_1) \tag{4.7}$$

where k_1 is the integration constant chosen to match a decay time equal to N clock cycles. In our experiments, $k_1 = 1/N = 1/2048$.

If this estimated phase shift $\tilde{\epsilon}$ exceeds a half of the FPGA clock cycle, we make a phase correction of the generated clock by one FPGA clock cycle. The correction is done by either decreasing or increasing the length of the next generated clock

cycle according to the sign of $\tilde{\epsilon}_i$ at the time, therefore the next rising edge of the generated clock is:

$$g_{i+1} = \begin{cases} g_i + \tilde{r}_i + 1, & \tilde{\epsilon}_i > 1/2 \\ g_i + \tilde{r}_i - 1, & \tilde{\epsilon}_i < -1/2 \\ g_i + \tilde{r}_i, & otherwise \end{cases} \tag{4.8}$$

where $\tilde{r}_i$ is a running estimate of the received clock period computed as an exponential moving average $\tilde{r}_i = (r_i - r_{i-1})k_2 + \tilde{r}_{i-1}(1 - k_2)$ is an exponential moving average of the received clock period. After the correction is done, the estimation of ϵ starts anew. We set $k_2 = 1/65536$, that corresponds to a 0.33 s averaging window. The window determines the time taken to track the frequency drift of the RFclock.

This synchronization process is shown in Figure 4.8. The phase error between the received master clock and the generated clock is accumulated over time. When the accumulated phase error exceeds the threshold, one period of the generated clock is corrected.

The described method provides the sampling clock frequency and phase synchronization. In addition, the absolute time counters at the modules also need to be synchronized in order to properly align in time the data collected at different modules. For this purpose, the data stream (RFdata) transmitted from the synchronization master is used. The data is transmitted with the bitrate of the reference clock, i.e. 200 kHz. The continuous data stream is logically divided into 1600-bit packets. Each packet has a header and a data payload. The header consists of a 13-bit pattern known as a Barker code of length of 13, followed by a 7-bit timestamp (Fig. 4.7). Because of its low-autocorrelation property, the Barker code is used to detect the start of the packet reliably even in poor radio conditions. Each slave module keeps track of the absolute time by counting received reference clock cycles, and also by comparing its own time counter with the timestamp sent by the synchronization master in the packet header. The data payload consists of a sequence of alternating 1's and 0's. The execution of the algorithm on the platform is described by Algorithm 2.

Algorithm 2: WHISPER synchronization algorithm

1: Synchronization master starts continuously transmitting the synchronization data sequence
2: Slave module recovers the master reference clock from the data stream and continuously estimates its period $\tilde{r}_i$
3: Slave module generates its own clock g_i with the period $\tilde{r}_i$ and measures its average phase shift $\tilde{\epsilon}_i$ with respect to the received reference clock r_i
4: Slave module compares the average phase shift $\tilde{\epsilon}_i$ with the threshold $\pm 1/2$ and updates the length of the next generated clock cycle g_{i+1} according to Eq. 4.8
5: The average phase shift $\tilde{\epsilon}_i$ is set to 0 and the step 2 repeats

4.3.3.2 Measurements

We show measurement results from experiments that evaluated two aspects of the synchronization algorithm. The first experiment measures the amount of synchronization between the generated clock of a single module and the received master clock. The second experiment measures the synchronization error between the generated clocks on four different modules. The error is measured as the phase shift between the clock of a reference module and the clocks of the other three modules.

The results from the first experiment are presented in Figure 4.9. We first show the empirical distribution of the deviation of the received clock period length from the master clock period length over $30 \cdot 10^6$ clock cycles in subfigure (a). Measurements are done for three conditions, namely in the absence of radio noise (clean), and in the presence of radio noise with two SNR levels (9 dB and 6 dB). The radio noise is generated by another RF transmitter for which we can set the transmission power and thus we can control the SNR. From these curves, we first see that the period deviation is quantized because the clock frequency of the RF module (6.4 MHz) is resampled using the faster FPGA clock which runs at 120 MHz. We also see that the means of the distributions are all around zero ($\mu^a_{clean} = 0.0001\%$, $\mu^a_{9dB} = 0.0002\%$, $\mu^a_{6dB} = 0.007\%$), while the standard deviations increase from ($\sigma^a_{clean} = 1.35\%$) in the clean case to ($\sigma^a_{9dB} = 1.37\%$) and ($\sigma^a_{6dB} = 1.40\%$) in the 9 dB and 6 dB SNR cases, respectively. The small probabilities associated with the tails of the distributions in the RF noise cases show the effect of the noise but do not compromise the synchronization.

Figure 4.9 (b) shows the distribution of the phase difference between the generated clock of the slave module and its received reference clock. The shape of this distribution is due primarily to the phase jitter of the received reference clock because the generated clock has substantially lower phase jitter (see Figure 4.11). Similarly to panel (a), the means of the distributions in panel (b) are around 0 ($\mu^b_{clean} = 0.058\%$, $\mu^b_{9dB} = 0.057\%$, $\mu^b_{6dB} = 0.070\%$), and the standard deviations of the probability distributions increase from ($\sigma^b_{clean} = 1.27\%$) for the clean case to ($\sigma^b_{9dB} = 2.38\%$) and ($\sigma^b_{6dB} = 5.96\%$) for the 9 dB and 6 dB SNR cases, respectively. The results show that, even in noisy conditions when the jitter increases, the synchronization algorithm is successful in keeping the average phase difference between the reference and generated clocks around zero.

From the same experiment, we measured the accumulated phase error between the received reference clock and the generated clock at one module (Figure 4.10). We can clearly see that in both clean (blue) and noisy condition (green for 9 dB and red for 6 dB), this accumulated phase error saturates quickly. The accumulated phase error curve for the clean case looks flat because the variance is significantly smaller ($\sigma_{clean} = 0.1\%$) than the variance for all the other cases ($\sigma_{9dB} = 2.3\%$, $\sigma_{6dB} = 4.5\%$, $\sigma_{gauss} = 8.7\%$). We compare these measurements with the output of a simulated scenario where the error distribution is the same as in the noisy condition with 9 dB SNR, i.e, a Gaussian distribution with $\mu = 0\%$ and $\sigma = 2.38\%$. In the simulation, the phase errors are independently drawn in sequence. In this case (gray curve in Figure 4.10), the accumulated phase error grows over time. This difference seen between the curves suggests that the errors in the received clock period of the hardware platform are not independent because probably the errors cancel out over

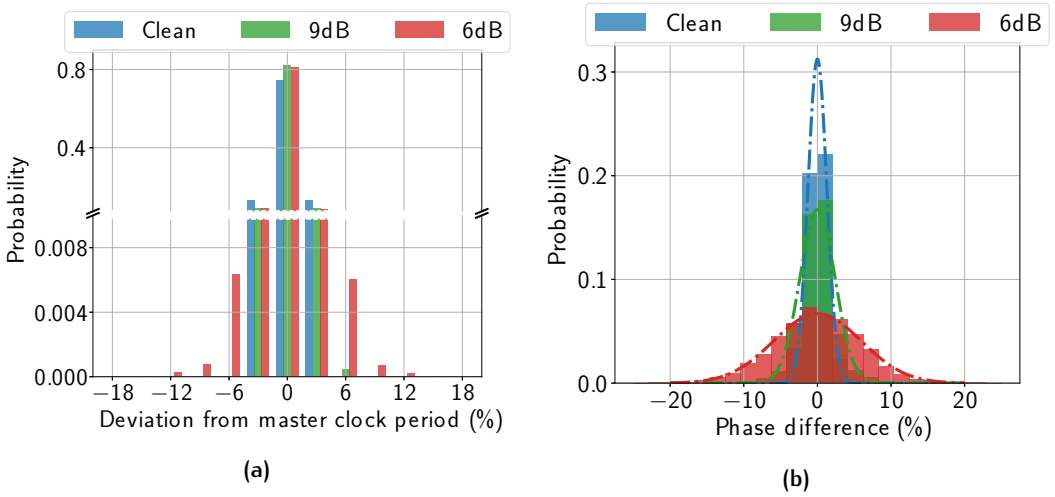

Figure 4.9: Deviation of period and phase of the received reference clock in three conditions: clean, RF noise with 9 dB SNR and RF noise with 6 dB SNR. (a) Deviation of the received clock period with respect to the master clock period. The y-axis is discontinuous in order to show the extremes of the distribution. (b) Phase difference distribution between received and generated clock signals.

time. This shows that even in noisy conditions the error does not grow indefinitely and can be corrected appropriately.

In the second experiment, we measure the level of synchronization between the generated clocks of four modules placed in different spatial arrangements from recordings of 1 minute. Figure 4.11 shows the measured phase shift and jitter of the clocks of three modules with respect to the clock of a reference module for four different spatial arrangements. The recordings are done in two rooms. The measured distributions in Figures 4.11 (a)-(c) come from a room with fewer RF-reflective surfaces than the room used for the measured data shown in Figure 4.11 (d).

For all spatial arrangements that we tested, the phase shift was less than 200 ns which is well within our specifications for precise localization and beamforming. The corresponding jitter, as measured by the standard deviation of the phase difference distribution, was less than 9 ns for the spatial arrangements used in Figures 4.11 (a)-(c). The increased jitter in Figure 4.11 (d) is very likely due to the increased RF-reflective material in the second room instead of the spatial arrangement of the modules.

The jitter value of around 9 ns (which is 0.025% of the 24 kHz sampling clock period) is low enough to guarantee that the sampling noise introduced by the jitter yields high SNR of the sampled signal (Higgins, 1996).

4.3.3.3 Functionality range

The results from the experiments in this section indicate that WHISPER is expected to work with most configuration of the nodes within a space of 3 m $\times$ 3 m, and in the presence of in-band radio noise down to a SNR of 6 dB. Below this value, the synchronization is not expected to work well, in particular, we assessed that

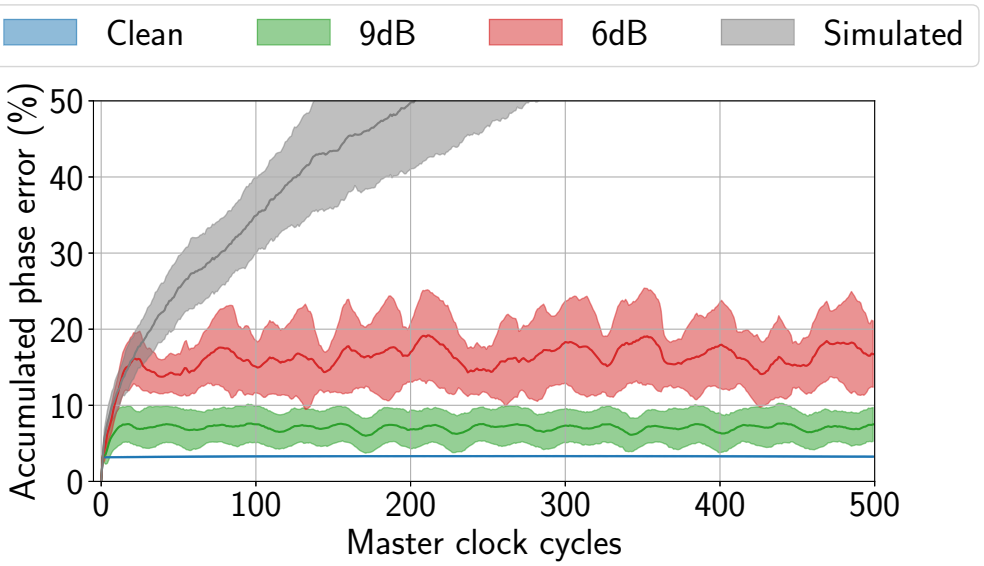

Figure 4.10: Accumulated phase error between generated clock and received clock for the simulated case of independent phase error (gray) and measured from the hardware platform in the cases where RF noise is absent (blue) or present with SNR of 9 dB (green) or 6 dB (red).

synchronization fails in the presence of 3 dB radio noise. The amount of RF reflective material such as large metal and concrete surfaces (ITU-R Recommendation, 2015) will also affect the synchronization quality. However, a synchronization signal on the WHISPER module indicates in a short time whether the system works in a chosen space.

Jitter and phase shift measurements for the first version of the synchronization algorithm are given in the Appendix A.3 Fig. A.25

4.4 BEAMFORMING EXPERIMENTS WITH WHISPER

4.4.1 Experiments with Prototype 1

We report results from a particular beamforming algorithm applied to recordings acquired with WHISPER in a conference room. The recordings were collected using two WHISPER-M4 prototype modules placed on a conference table in front of two loudspeakers as shown in Fig. 4.12. This configuration had eight SPI microphones, four from each module. The recordings were done over a period of 1 min with either two male or two female speakers reading audiobooks from two different loudspeakers. To investigate effects of different SNR, we kept the volume of one speaker constant while we varied the volume of the second speaker. We also investigated the effect of placing a microphone close to either the desired source or the interfering source. We kept the SNR at 10 dB and recorded a scenario where one of the microphones was 20 cm from the loud source and a second scenario where the microphone was 20 cm from the soft source.

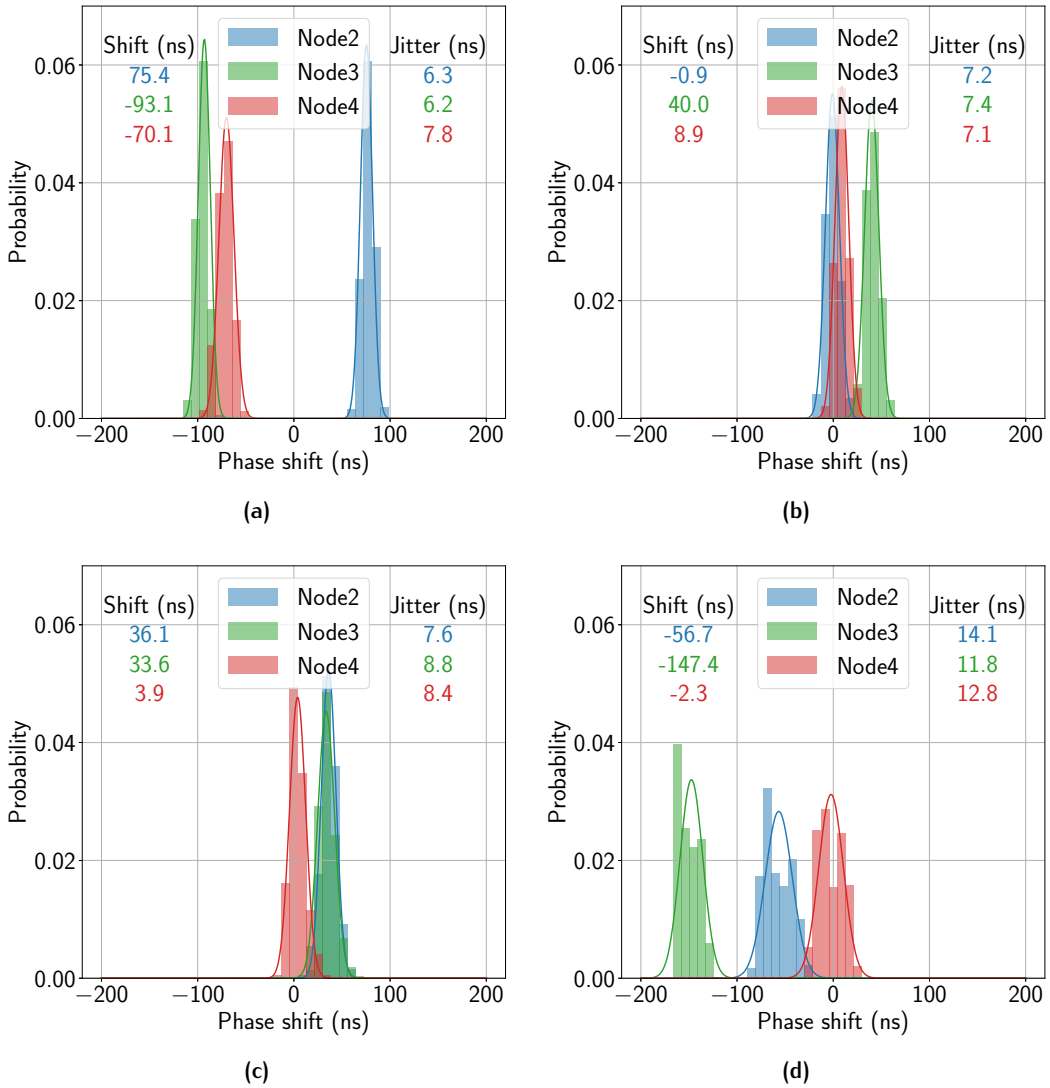

Figure 4.11: Distribution of phase differences between the clock of a reference module and the clock of other three modules for four spatial configurations. (a) Square arrangement with an edge length of 2.5 m. (b) Line arrangement with a spacing of 0.8 m between modules. (c) Random arrangement where all modules are equidistant (1 m) from the transmitter. (d) Random arrangement within a radius of 2 m. The data for panels (a), (b) and (c) was recorded in a room with fewer RF-reflective surfaces than the room in which the data for panel (d) was recorded. The mean ('Shift') and standard deviation ('Jitter') of the distributions are reported within the panels.

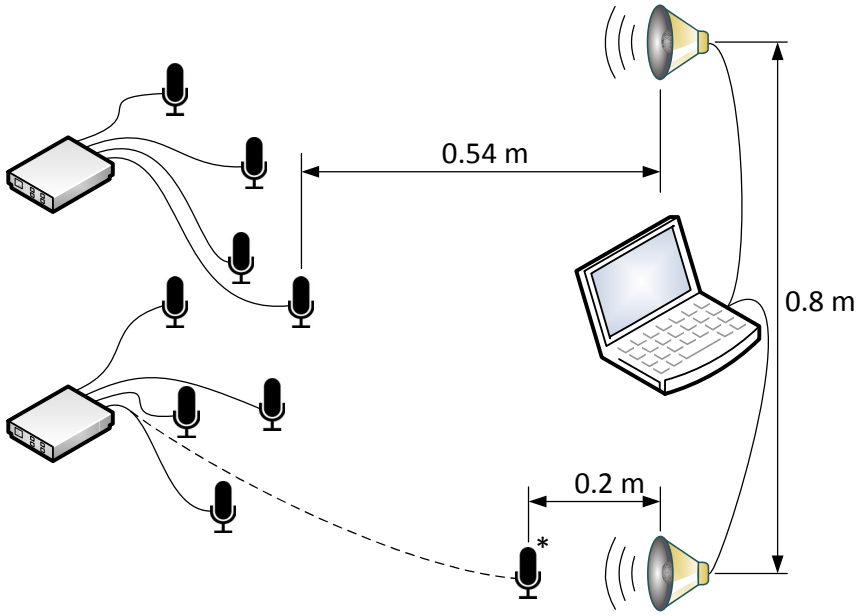

Figure 4.12: Experimental setup for the WHISPER dataset. Approximate relative positions of speakers and microphones are shown. Data are transmitted to the laptop wirelessly via Wi-Fi. The *-marked mic position is used for "10 loud" and "10 soft" experiments.

We used the Linearly Constrained Minimum Variance (LCMV) algorithm (Frost, 1972; Van Veen *et al.*, 1997) for our experiments. It is a spatial filtering method widely used for speech enhancement. We used this algorithm to localize the sound sources based on data coming from the microphones.

The basic idea of LCMV is to minimize the power of the filtered signal obtained by summing the signal from each microphone after applying meaningful delays, with a constraint. This constraint forces the filter to output power from a specific location and the power minimization allows one to suppress the signals coming from undesired locations.

To estimate the constraint matrix, we recorded additionally each speaker alone for 15 seconds. The details of the algorithm applied for this setting are described in (Kiselev, Ceolini, *et al.*, 2017b).

The beamforming algorithm was applied in real-time on the data recorded from two modules when the microphone sampling frequencies were either synchronized or not synchronized across the modules. The results were obtained using a sampling rate of 8 kHz and 256 samples in a window.

We evaluate the performance of the LCMV beamformer using two metrics: the Signal-to-Interference Ratio (SIR) metric which is a classical evaluation measure for blind source separation (Vincent *et al.*, 2006) and the Short-Time Objective Intelligibility (STOI), a measure of speech intelligibility. The SIR value is calculated as described in (Vincent *et al.*, 2006) and in Eq. 4.9. For this, we need to provide both

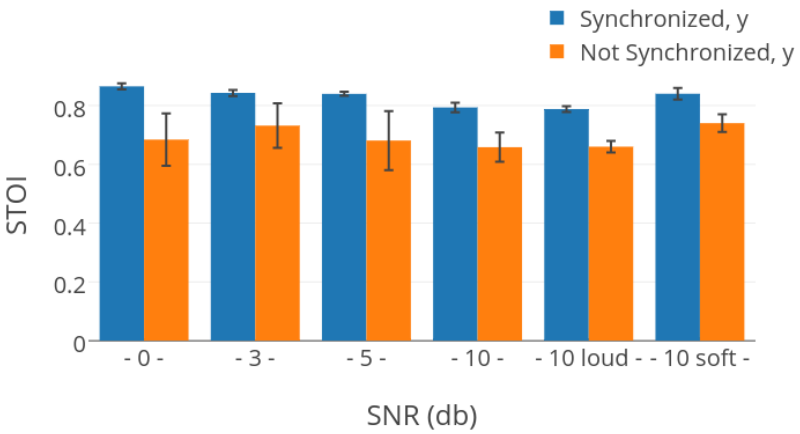

Figure 4.13: STOI scores computed from synchronized and non-synchronized samples for four different signal-to-noise ratios of the microphone placement shown in Fig. 4.12 and by placing a microphone close to the speaker with the louder source ("10 loud"), and then to the speaker with the softer source ("10 soft").

the separated signals $y_{target} = w_{target} \cdot x$ and $y_{interf} = w_{interf} \cdot x$ which are obtained by estimating speaker dependent weights w_{target} and w_{interf},

$$SIR = 10 \cdot log_{10} \frac{||y^2_{target}||}{||y^2_{interf}||} \tag{4.9}$$

The STOI metric evaluates the quality of a speech signal with respect to ground truth. The score ranges from 0 to 1 where 1 corresponds to a perfect separation (or denoising) in which the separated signal from a mixture is as intelligible as the original signal.

Figures 4.13 and 4.14 show the beamforming quality through STOI and SIR scores of the separated signals when using either synchronized or non-synchronized samples of the audio mixture recorded with two WHISPER-M4 prototype modules in a conference room.

In particular, Fig. 4.13 shows that the average STOI score for synchronized samples is almost 20% higher than the score for non-synchronized samples. Fig. 4.14 shows that the SIR scores for non-synchronized samples are not statistically better than the SIR obtained by selecting the microphone with the best SNR out of all the eight microphones. Both figures show the degradation in the quality of the separated source when the samples are not synchronized. Additionally, we can see that having a microphone very close to either the desired source or the interfering source is beneficial for the beamforming. In particular having a microphone very close to the desired source, which is 10 dB softer then the interfering source increases the STOI by 3% and the SIR by 15%.

The results presented in this section are acquired with the first prototype of the WHISPER platform, which has 12-bit SPI microphones, 8 kHz sampling rate and higher jitter of the sampling clock – 150 ns vs 9 ns in the final version. However, the long-term stability of the phase synchronization is already achieved in this

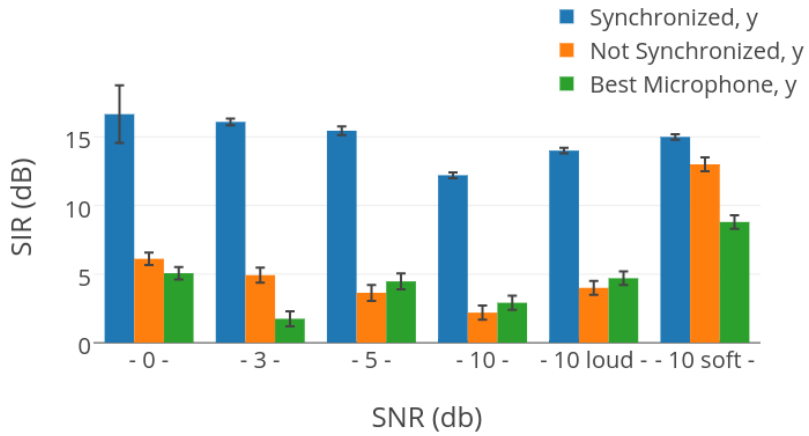

Figure 4.14: SIR scores computed from samples which are synchronized and non-synchronized for four different signal-to-noise ratios, compared with SIR obtained from the best microphone, and by placing a microphone close to the speaker with the louder source ("10 loud"), and then to the speaker with the softer source ("10 soft").

prototype, making the beamforming possible. Improved jitter, sampling rate and resolution in the final version of the platform give better overall sound quality at the output.

4.4.2 Speech Separation Experiments with final WHISPER

The previous experiments above were conducted using the first prototype of WHISPER. The final WHISPER platform was used to benchmark multi-channel speech separation algorithms in the wild (Ceolini, Kiselev, *et al.*, 2020). This paper which also included the details of the multi-channel platform, describes the speech separation experiments that were carried out using this platform. The author together with his co-author, E. Ceolini, created the experimental setup by placing four WHISPER modules at random positions around a conference table in a normal conference room. The synchronization and desynchronization of the modules were checked by both before the recordings were started. The data analysis on the beamforming experiments was carried out by the co-author. The results from one of the speech experiments are shown in Fig. 4.15. In this experiment, a single speaker is extracted from a mixture of speakers (considered as noise). The speech of the single speaker is extracted for three SNR levels of $[0, -5, -10]$ dB. From this figure, we see the benefits of increasing microphones from the increasing measured STOI. The trend is similar for all three SNR conditions. Beamforming with the ad-hoc arrangement is possible because of the synchronization algorithm described in the main text of this chapter.

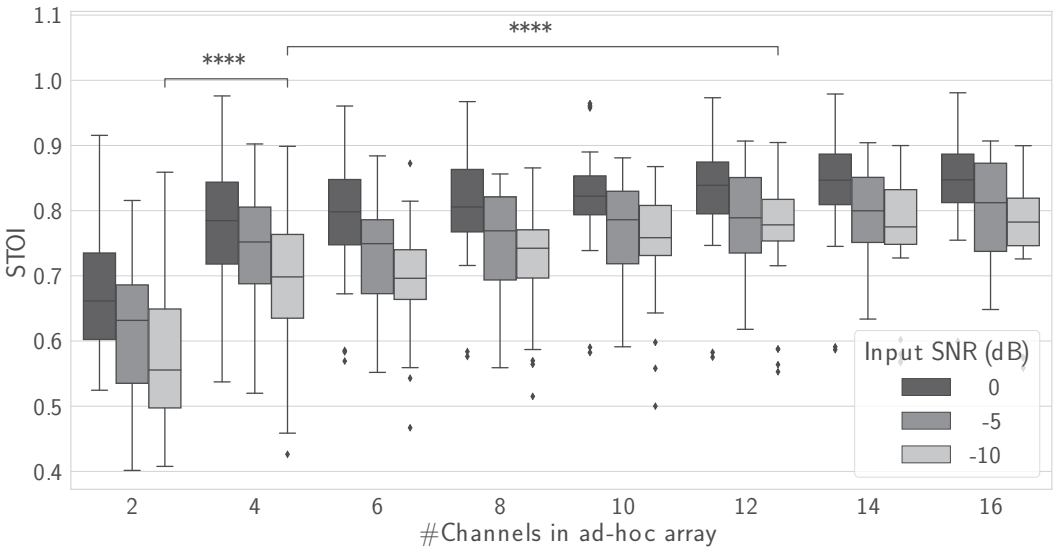

Figure 4.15: STOI of enhanced speech in a speech enhancement scenario for an increasing number of microphones and tested across three different input SNRs of speech in noise. Adapted from (Ceolini, Kiselev, *et al.*, 2020).

4.5 DISCUSSION

This work describes a distributed portable multi-microphone platform with capabilities for acting as nodes in a wireless acoustic sensor network. This platform addresses three main challenges for use in a WASN. The first is the synchronization of the sampling clocks across different modules with 200 ns precision and a clock jitter of less than 9 ns rms. The second is the synchronization of data packets received from multiple modules and the third is the development of a wireless transmission system that will ensure low latency transmission between the nodes of the platform. Our beamforming results from the microphone samples in the WHISPER dataset demonstrate that this tight synchronization of the sampling clock on all modules is important for obtaining the best beamforming results and that beamforming with this platform gave SNR results that are in line with the expected outcome of the algorithm. The platform can be miniaturized in the future to further reduce both the latency and power consumption, and keeping only the features that are needed for the proposed solutions in this work.

5 DISCUSSION AND CONCLUSION

5.1 SUMMARY

In this thesis, we developed hardware test beds that allow us to explore event-based sensory processing algorithms and regular sampling based algorithms in real-world conditions. The goal of the thesis was three-fold: 1) to develop a hardware test bed for implementing spiking networks together with spiking sensors to study a possibility of using multiple sensors of different modalities to improve classification performance in a real-world conditions; 2) to implement a local automatic gain control mechanism to increase the input dynamic range of a spiking cochlea operating in natural environments where the sound dynamic range can be greater than 60 dB; 3) to implement a multi-microphone hardware platform that can be used for real-time beamforming as part of a wireless acoustic sensor network.

SPIKING SENSOR FUSION SYSTEM In Chapter 2, we created a complete spiking deep network accelerator (n-Minitaur) that was capable of processing spikes from a DVS and a DAS.

The n-Minitaur system achieved real-time classification at event rates up-to 8.6 kEvents/s on the network with $\sim 700,000$ synapses. However the event rate from the sensors in a real environment in a MNIST recognition task was 5X higher than the network could process in a real time, so we had to implement spike decimation by a factor of 5 for real-time processing. The network was hardly affected even with this decimation. Our experimental results with the hardware sensory fusion system showed that the classification accuracy of the network in a digit recognition task increased when input events from two sensors of different modalities were used. The system at the time (2016) showed low-latency responses in the order of 5 ms (time between 1st input spike and 1st output spike) in a visual classification task, which is suitable for robotic and fast vision-based control applications.

In short, we showed a first demonstration of a real-time hardware FPGA system that implemented an event-driven spiking network receiving input spikes from both a Dynamic Vision Sensor and a Dynamic Audio Sensor. This demonstration completed in 2016 was the first study to use sensor fusion from spiking audio and video sensors to improve classification accuracy of spiking deep neural networks in noisy conditions. The latency can be improved even further with the availability of newer embedded platforms. Since then, there are a few more examples of systems that combine both spiking sensor modalities, e.g. for scene stitching around an active talker with the sensors mounted on a pan-tilt unit (Klein *et al.*, 2015), a robot navigation task (Chan, Jin, *et al.*, 2012), and word recognition using a lip-reading spike dataset (Li, Neil, *et al.*, 2019).

AGC One observation of the experiments with n-Minitaur is that there is a wide range of spike rates from the spiking cochlea when the sound volume changes. With the wide range of sound amplitudes in natural environments, the network will not perform well because of the changing spike rates. Because the local AGC model was difficult to implement in the ASIC spiking cochleas, we developed in Chapter 3, an alternate system-level local AGC mechanism that uses the output spike rates of individual channels. This mechanism maintains a more constant spike rate per channel across a wide range of input amplitudes.

The algorithm as implemented on an FPGA, increases the range of acceptable input signal amplitudes for the DASLP cochlea by 32 dB and allows the cochlea to operate over an input dynamic range of 54 dB while maintaining a bounded event rate from the individual cochlea channels. We demonstrated that the output of the AGC-controlled cochlea led to a better accuracy in a task of speech versus noise classification when the sound volume is different in the training and the testing datasets.

This work is the first reported implementation of an automated local gain control mechanism that uses the spike outputs of the frequency-specific filter channels of a Dynamic Audio Sensor spiking cochlea.

MULTI-MICROPHONE DISTRIBUTED PLATFORM FOR AD-HOC NETWORK Chapter 4 presented the work carried out this thesis to develop a real-time distributed multi-microphone hardware platform for a wireless acoustic sensor network. This platform was developed for evaluating speech separation algorithms in the wild. It was used within a multi-partner European project COCOHA as a platform for general audio scene analysis to combine audio speech separation algorithms with EEG recordings form an active listener in order to stream the attended speech in a cocktail party scenario to a hearing impaired person.

The results showed that we could construct a portable system which can be powered on a 5 W battery power source and has the needed logic resources to implement the algorithm to synchronize the sampling clocks across modules and to do some simple on-board processing. Although the prototype of the WHISPER platform provided decent sampling frequency and phase synchronization quality (Kiselev, Ceolini, *et al.*, 2017b), we had to develop a second prototype because the sampling clock jitter on the first prototype was too high, introducing an audible noise to the output audio signal. In order to eliminate this problem the synchronization algorithm we improved the synchronization algorithm of the 1st prototype and designed custom microphones based on digital microphones from InvenSense.

The system was used in a set of experiments in different rooms to demonstrate that it was useful for evaluating beamforming and speech enhancement algorithms in the wild (Ceolini, Kiselev, *et al.*, 2020).

5.2 OUTLOOK

Hardware spiking cochleas (e.g. the Dynamic Audio Sensor (Liu, Schaik, *et al.*, 2014)) and software cochlea models are being tested in recent years as front-ends for

audio tasks such as source localization (Anumula, Ceolini, *et al.*, 2018; van Schaik *et al.*, 2009), speech recognition(Wu *et al.*, 2018), speaker verification, multi-modal recognition (Kiselev, Neil, *et al.*, 2016a) and keyword spotting (Ceolini, Anumula, *et al.*, 2019). Spike features such as constant time bin and constant spike count features (Acharya *et al.*, 2018; Anumula, Neil, Delbruck, *et al.*, 2018) are usually presented as inputs to a machine learning classifier such as the SVM classifier. A few studies show that using such features instead of conventional features such as log-filter banks or Mel Frequency Cepstral Coefficient (MFCC) features can lead to similar accuracies for a speech recognition task. Some studies show the usefulness of the spike outputs, e.g. the accuracy of a word recognition task using spiking cochlea inputs, decreases at a slower rate with decreasing signal-to-noise (SNR) input (Uysal *et al.*, 2007; Zai *et al.*, 2015).

Deep neural networks (DNNs) such as recurrent neural networks have been applied quite successfully on the spiking cochlea outputs for various audio classification tasks (Anumula, Neil, Delbruck, *et al.*, 2018; Neil *et al.*, 2016) including a demonstration of a hardware system with a DAS and DNN hardware in the real world (Gao, Braun, *et al.*, 2019; Kiselev, Neil, *et al.*, 2016a).

The n-Minitaur event-driven neural network hardware system in Chapter 2 can also be used to implement event-driven deep networks that have been trained to perform simple speech recognition using the spiking cochlea input (Neil *et al.*, 2016). The trained deep networks can be converted into spiking networks with conversion methods that use rate codes (Rueckauer, Lungu, *et al.*, 2017) or temporal codes (Liu and Douglas, 2004; Rueckauer and Liu, 2018).

The n-Minitaur system can only be interfaced to a maximum of 3 AER sensors without a PC in the data path. However the PC is still required for uploading the network structure and coefficients and for configuring the sensors. The need for a PC can be eliminated by using modern SoC FPGAs, such as Xilinx Zynq Ultrascale+, which has on-chip CPU cores.

The AGC algorithm in Chapter 3 can be enabled in the spiking cochlea and then used to see if the performance accuracy will increase for networks trained on audio tasks such as keyword spotting. For example, the impact of the AGC-controlled cochlea can be determined for an audio classification task such as digit recognition (Ceolini, Neil, *et al.*, 2016; Neil *et al.*, 2016; Uysal *et al.*, 2008),speaker verification and keyword spotting (Ceolini, Kiselev, *et al.*, 2019).

The implemented AGC algorithm on FPGA can also be incorporated into real-time FPGA hardware network accelerators e.g. EdgeDRNN (Gao, Braun, *et al.*, 2019; Gao, Rios-Navarro, *et al.*, 2020) with spike information that includes the local gain setting. Other approaches can be investigated in the future for implementing the AGC control loop for e.g. through continuous spike integration.

The WHISPER platform in Chapter 4 can be minitaurized by using custom PCB with Zync ultrascale (with FPGA and processor), and be extended to process other sensor modalities. It can also be interfaced to the Dynamic Audio Sensor and would allow future studies of using regular-sampled audio input versus the asynchronous outputs of the cochlea for a specific audio task such as speech recognition, keyword spotting.

A APPENDIX

A.1 SUPPLEMENTARY MATERIAL TO CHAPTER 2

A.1.1 n-Minitaur Implementation Details

Fig. A.1 shows logical interfaces of the USB clock domain decoupler, which provides data transfer between USB2 (IFCLK = 48 MHz) and FPGA (sys_clk = 105 MHz) clock domains as described in Sec. 2.3 (depicted as USB FIFO in Fig. 2.7). The USB2 interface chip FX2[1] has two FIFOs: IN and OUT, which share some control signals (nSLRD — Slave Read, active 0; nSLWR – Slave Write, active 0). The FIFOADR[1:0] signal defines which FIFO is currently controlled by these signals and connected to the data bus FD[7:0]. The USB decoupler FSM shown in Fig. A.2 controls data transfer between these external FX2 FIFOs and the internal Xilinx dual-clock FIFOs usb_wr_fifo and usb_rd_fifo. The right-side interfaces of these two FIFOs together with the WR_DONE_REQ and WR_DONE_ACK signals constitute an internal I/O interface, which is connected to the Command Decoder in Fig. 2.7. WR_DONE_REQ and WR_DONE_ACK signals provide a way to initiate a USB transaction when the FX2 OUT FIFO buffer is not filled yet by issuing a nPKTEND signal.

An example of the state transition diagram with all essential input, output and internal signals is given in Fig. A.3. It shows read and write operations of the external FX2 FIFOs, and read-to-write and write-to-read transitions. An IDLE state for one IFCLK clock cycle is added when switching from READ to WRITE state in order to avoid bus contention on the FD bus.

1 https://www.cypress.com/file/138911/download

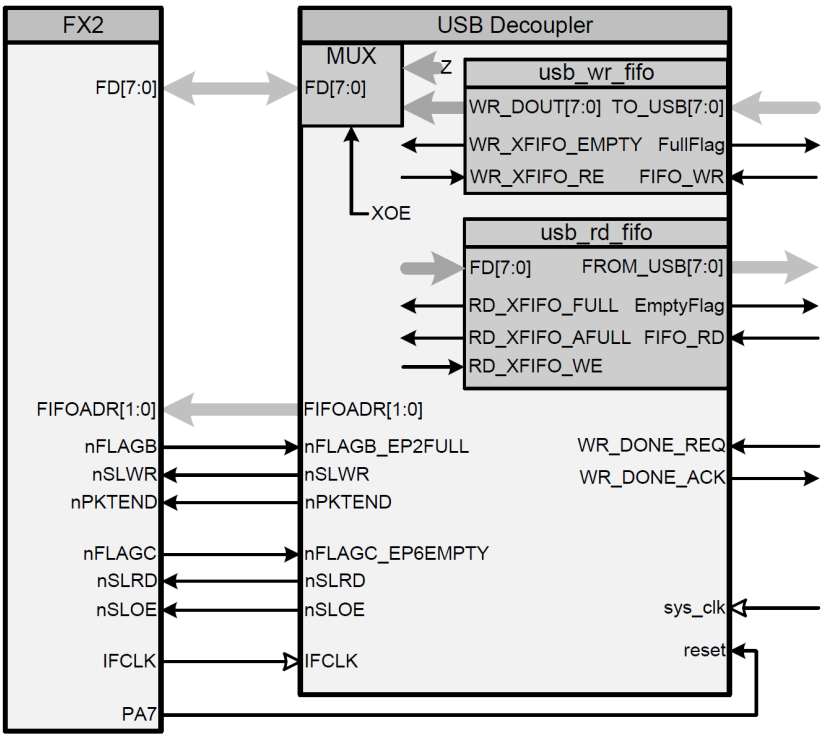

Figure A.1: Interfaces of the USB Decoupler to the FX2 and to the rest of FPGA.
Notes:
1. FIFOADR affects only nSLRD and nSLWR signals;
2. FLAGB and FLAGC are preassigned to EP2FULL and EP6EMPTY;
3. First Word Falls Through (FWFT) FIFO is used. The FWFT feature adds two clock cycle latency to deassertion of the Empty flag, when the first data is written into an empty FIFO;
4. Initiating a write transaction to the FX2 FIFO when it is full is not destructive to the contents of the FIFO;
5. nSLWR signal is not registered because it is just an inversion of the registered signal WR_XFIFO_RE and it has just 10.4 ns setup time, which gives 10.4 ns slack time at 48 MHz clock.

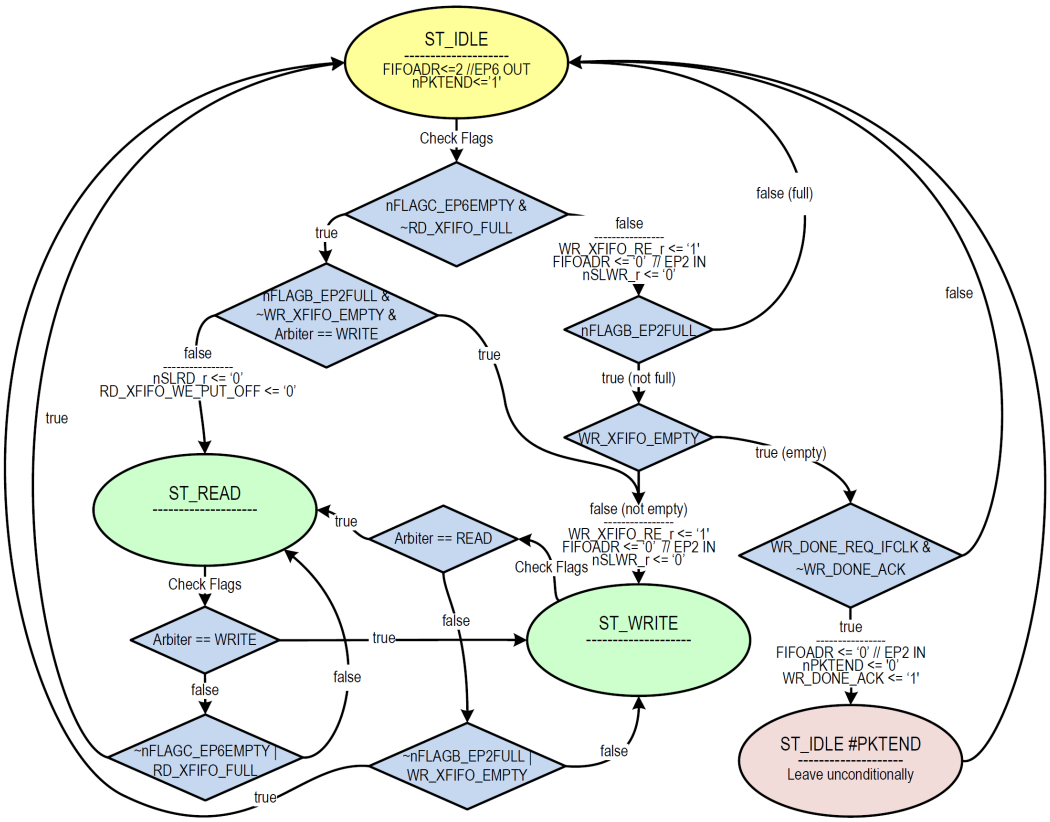

Figure A.2: USB decoupler FSM state transition diagram.

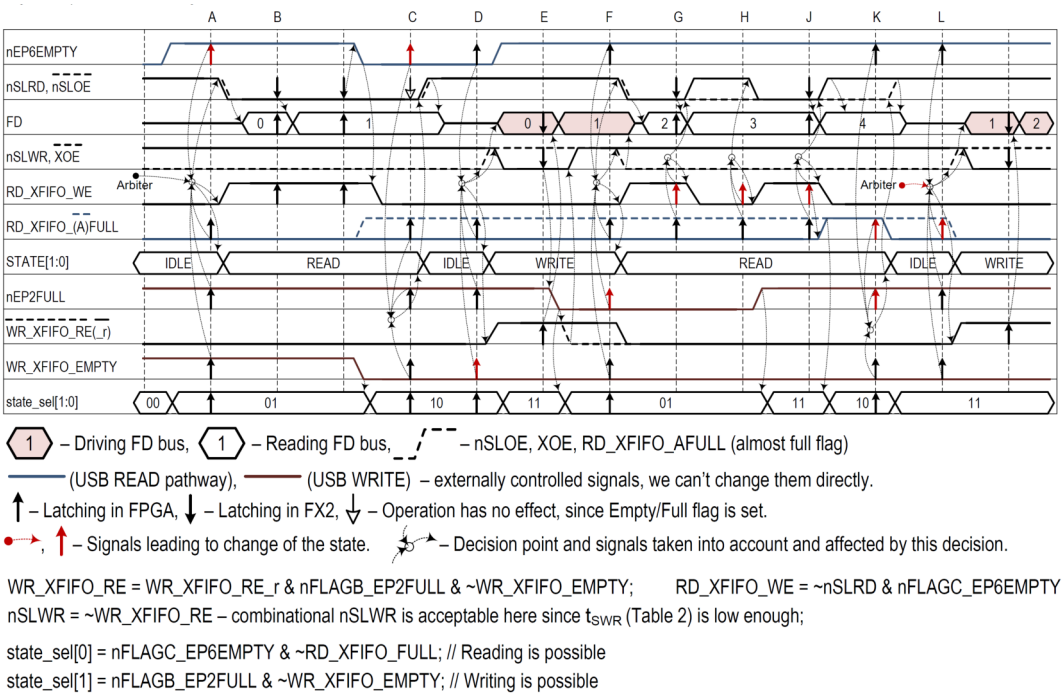

Figure A.3: An example of the FX2 FIFO control signals during transitions between IDLE, READ and WRITE states. In particular, the cases when the read FIFO is empty (C, D) and the write FIFO is full (F, G, H) are shown. The dashed vertical lines represent the IFCLK 48 MHz clock.
Description of the States:
A) Enter READ state by nEP6EMPTY low-to-high transition;
B) Read 2 bytes (FD = <0, 1>) from FX2 FIFO until the read FIFO is empty;
C) Leave READ by nEP6EMPTY and go to IDLE first, to prevent a bus-contention on the FD bus;
D) Go from IDLE to WRITE triggered by WR_XFIFO_EMPTY;
E) Write 1 byte (FD = <0>, red) to FX2 FIFO. Byte <1> (red) is not written here;
F) Leave WRITE by nEP2FULL and go to READ by nEP6EMPTY immediately, since there is no risk of bus-contention here;
G) Read 1 byte (FD = <2>) from FX2 and pause reading (nSLRD = 1, RD_XFIFO_WE = 0) since *RD_XFIFO* is "Almost Full";
H) Handle the case when RD_XFIFO_FULL does not go high after writing one byte to a FIFO with RD_XFIFO_AFULL = 1;
J) Proceed with reading by 1 byte (FD = <3, 4>);
K) Leave READ by RD_XFIFO_FULL and go to IDLE;
L) At this point both operations (READ and WRITE) are possible, so we go to the WRITE state by decision of the Arbiter and write bytes (FD = <1, 2>, red).
Note: at the states E, J, L+1 the next state also depends on the Arbiter. It could change the state at any of these points since both state_sel bits are set to 1.

A.2 SUPPLEMENTARY MATERIAL TO CHAPTER 3

A.2.1 Input Signal Conditioning Circuit

The DASLP chip has a differential input and the DASLP board has a uni-polar power supply, however, most of the PC sound cards have single-ended AC-coupled audio output. Hence, a signal conditioning circuit is required in order to feed the chip with the audio signal from the PC. The simplified signal conditioning circuit is shown in Fig. A.4. The resistor divider R1−R2 together with resistors R3−R4 sets the input common mode voltage to approximately 0.6 V when the input signal level is 0 V. A differential unity-gain buffer (DA1) is used to remove the common-mode signal and to set the output common-mode (OCM) voltage to the operating point (250 mV) of the Cochlea chip (DA2). The differential ADCs DA3 and DA4 are used to measure the amplitude of an input signal and the amplitude at the output of the filter-amplifier of one of the cochlea channels.

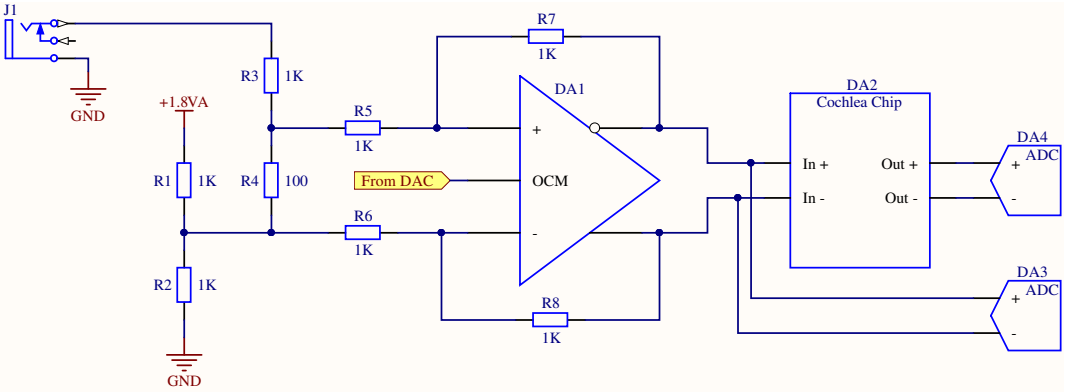

Figure A.4: A simplified diagram of a circuit for conversion of the single-ended audio output to the differential Cochlea input.

A.2.2 Calibration of Input from Sound Card

We calibrated the output of the audio card in the range of $40 - 10000$ Hz using the onboard 18-bit ADC. We apply the inverted frequency response to all test signals generated later. Next, we tested linearity and flatness of the frequency response at different amplitudes. The adjusted frequency responses were flat in the frequency range of interest $50 - 8000$ Hz at all amplitudes in the range of $1 - 100$ mV (Fig. A.5). The audio samples were played through the same sound card without frequency response correction.

A.2.3 DASLP FPGA Control Logic Structure

The DASLP control logic is implemented on the Lattice FPGA LFE3-70EA. The FPGA communicates with a PC through the Cypress FX3 chip (CYUSB3014-BZX) that implements USB 3.0 interface. There are two interfaces between the FPGA and FX3 chip – a high speed FIFO interface for data transfer from the cochlea chip to a

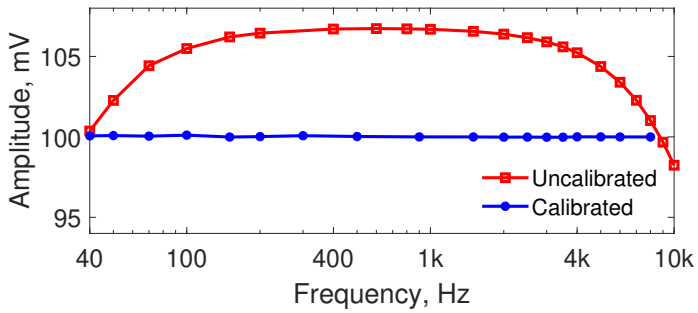

Figure A.5: Uncalibrated and calibrated output of the sound card.

PC and a slower SPI interface for configuration of all FPGA modules and setting on-chip and off-chip bias values A simplified diagram of the FPGA logic is shown in Fig. A.6.

A.2.4 Implementation of ADC Data Transmission over Asynchronous AER Bus

In order to observe the signal waveform that elicited a series of spikes at the output of a cochlea channel; and to be able to compare the input and output signals of the bandpass filter-amplifier, we use two ADCs that are connected to the input and multiplexed output of the filter bank. ADC samples are taken synchronously at regular time intervals and read out to an FPGA via an SPI interface. Since the ADC sampling frequency is generated from the same clock signal that is used for the cochlea events timestamping, there is no desynchronization of the cochlea events and the ADC samples possible. We considered two possibilities for transmission of the ADC data to a PC for further processing – setting up a separate USB endpoint for the ADC data and merging the ADC data into one stream with the cochlea events. We found the latter option to be preferable, since it requires no hardware or firmware modification in the connection between the FPGA and the USB FX3 chip and it also makes possible transmission of the ADC samples without using USB to any other processing device that has an AER interface.

However, the selected approach, if implemented straightforwardly, would have a problem because the ADC samples the signal at the "Start of Conversion" time (Fig. A.7), but the data can be acquired and send to the AER bus only after "ADC Conversion" period, which is about 2 μs. In addition, all ADCs have to be read sequentially over common SPI bus, that takes another 1 μs per ADC channel. For two-channel ADC these delays sum up to 4 μs delay between the start of conversion and sending the data samples. There could be a few cochlea spikes generated and sent to AER FIFO within this period (see the cochlea spikes on Fig. A.7 within $0-4$ μs interval). In this case, the order of ADC and cochlea events would be reversed and hence, correct comparison of the cochlea events and waveforms would be difficult.

In order to avoid this problem, we issue a Special Event (subcode x"2C", Fig. A.8 top) at the beginning of the ADC Conversion cycle. This event denotes the time when the signal was sampled by the Sample-and-Hold circuit, and the subsequent data samples are attributed to this time. The data samples are also transmitted as

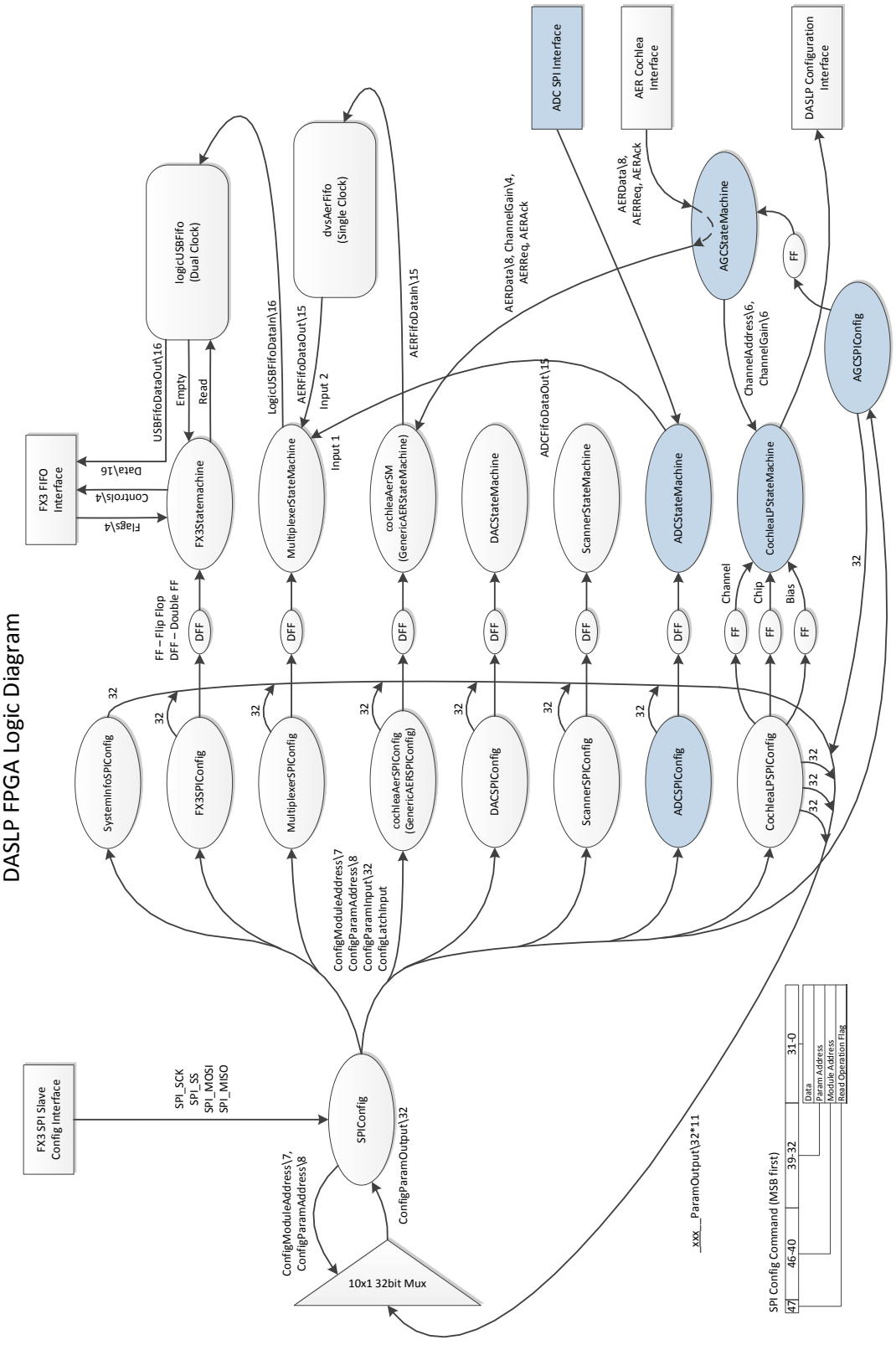

Figure A.6: FPGA logic architecture for the DASLP cochlea. The shadowed blocks are designed or significantly modified within this work.

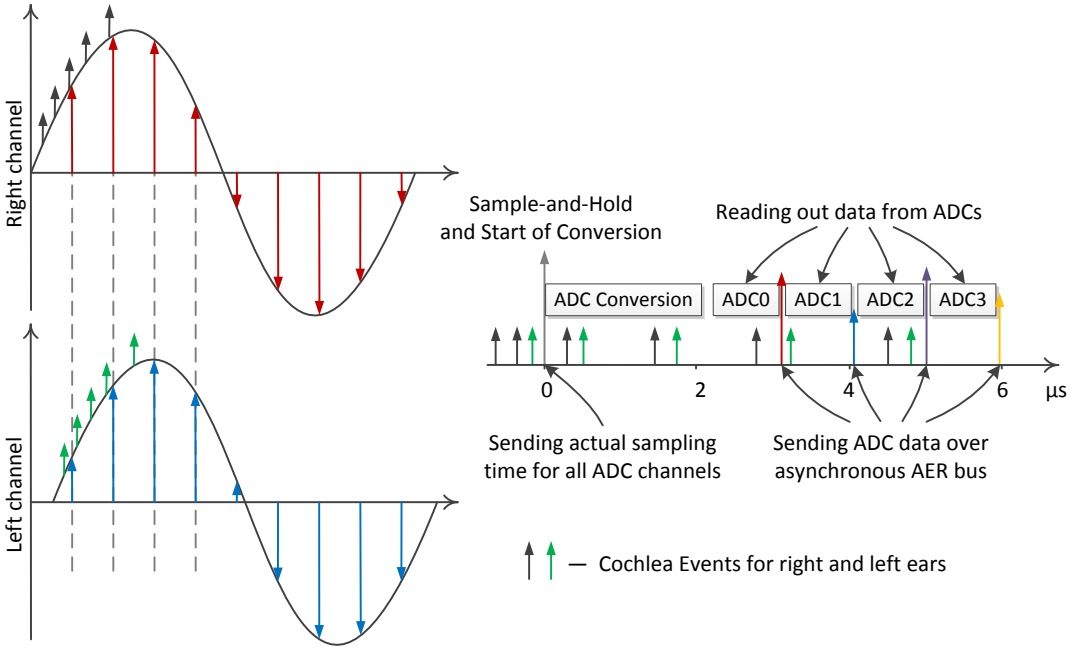

Figure A.7: Implementation of ADC data transmission over asynchronous AER bus.

special events (Fig. A.8 bottom). These special events have a separate Event Code b"100" and can carry 12-bit data payload. Since the ADC samples are 18-bit, two events are required to transmit one ADC sample. The ADC sample is split into two 9-bit parts – MSB and LSB. Bit 9 of the ADC Data Special Event is set to "1" for "MSB" and to "0" for "LSB". The remaining two bits (11:10) of the event data payload define the ADC channel number. The ADC finite state machine (FSM) is designed to support up to four ADC channels, however, only two ADCs are available on the DASLP board – #0 and #1.

The data samples from both ADCs are timestamped, labeled with a "Special Event ADC Data" bit code and injected into the event data stream from the DASLP chip. Then data packets comprising both the output spike data and input/output signal ADC samples are written to OUT-endpoint FIFO of the FX3 chip.

The ADC FSM supports two sampling frequencies – 16 kHz and 44.1 kHz, the same frequency is used for all the channels. The frequency can be selected in the jAER software. The detailed diagram of the ADC control FSM is given in Fig. A.9.

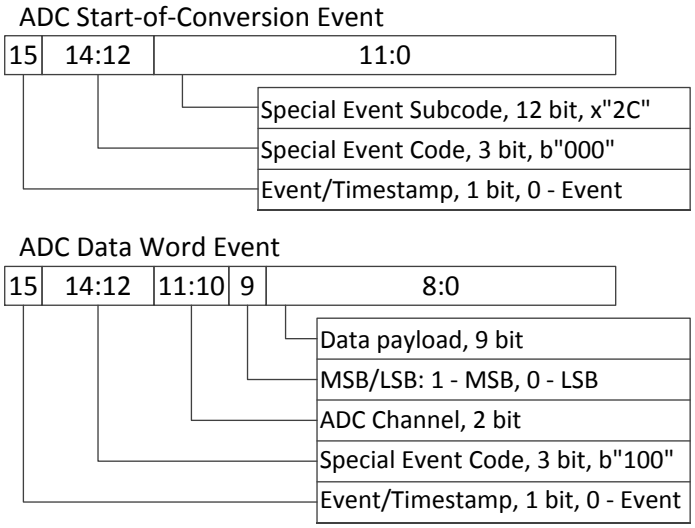

Figure A.8: Format of AER Special Events for ADC data transfer over AER protocol.

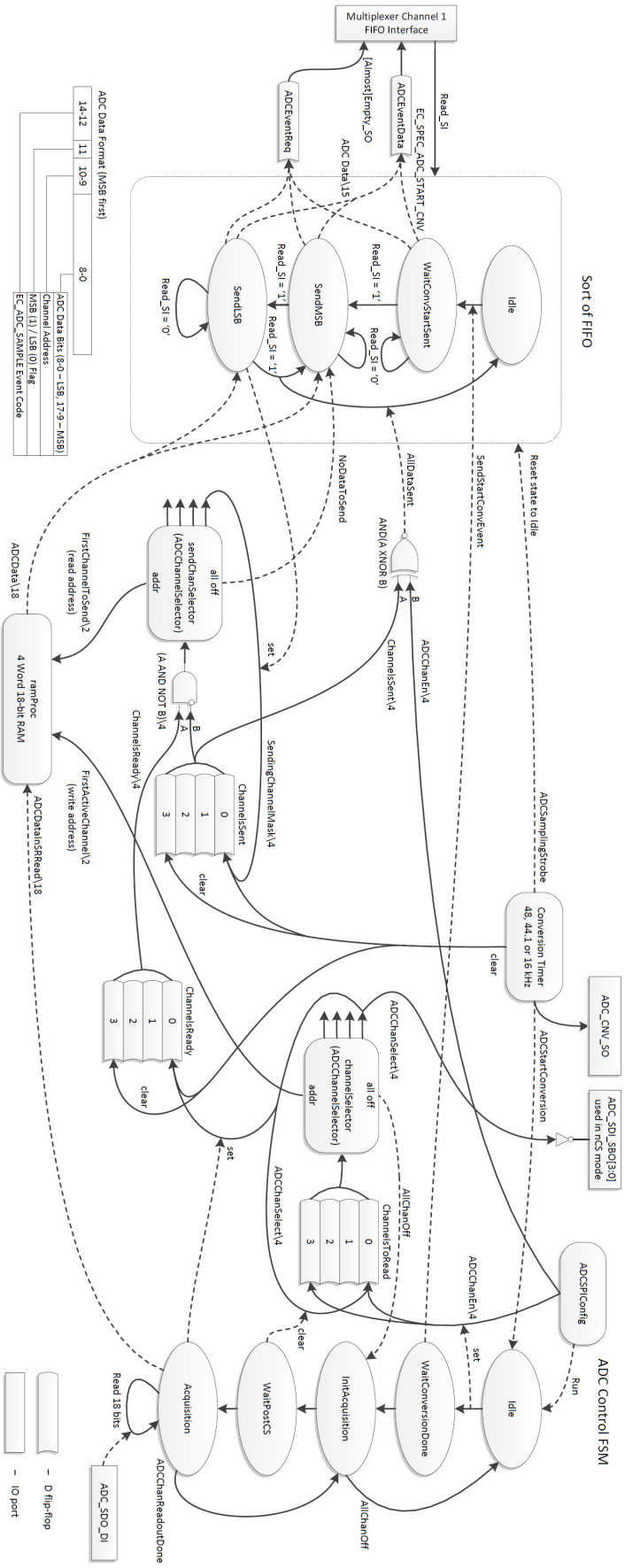

Figure A.9: Four channel ADC control state machine

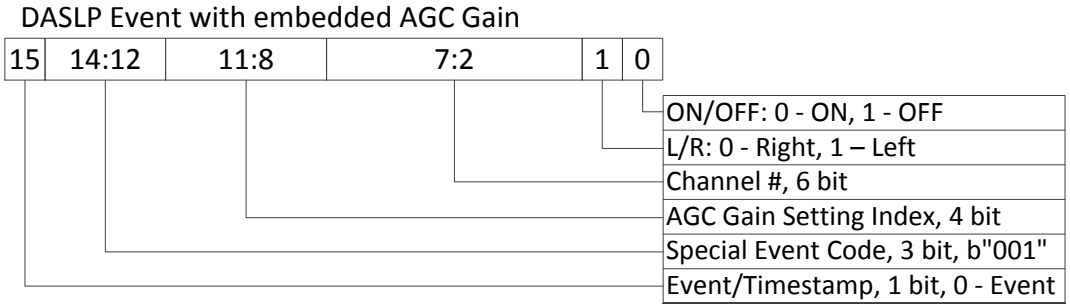

Figure A.10: DASLP cochlea event bit structure.

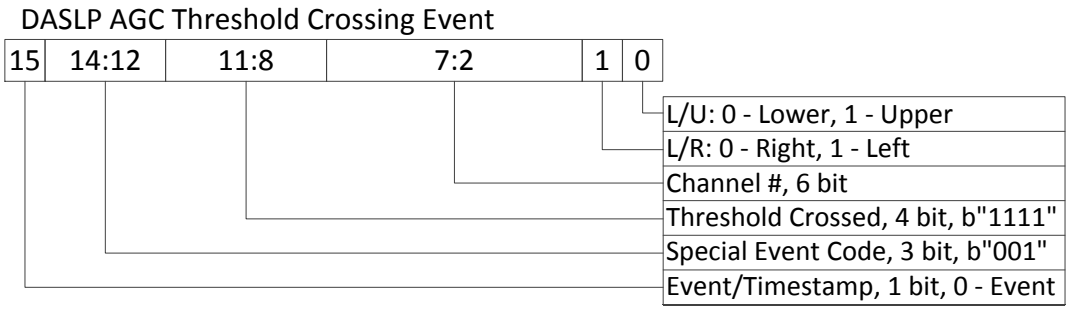

Figure A.11: DASLP AGC Threshold Crossing (or Gain Update) event bit structure. Bit 0 indicates which threshold is crossed, bit 1 — the ear currently used for AGC control. When the bits [11:8] are set to b"1111" the event is considered to be a threshold crossing event, otherwise it is a conventional cochlea event (see Fig. A.10).

A.2.5 Embedding the AGC Channel Gain Information into DASLP Events

Figure A.10 shows the structure of a modified DASLP event, which carries information about the current gain of a channel at the moment when the event was generated. Bit 15 is set to '0' indicating that this is an event. Bits [14:12] when set to "001" indicate that this is the "Y Address Event". Four bits [11:8] define the current gain of a channel (gain index values form 0 to 11 are valid). If these bits are equal to "1111" (15), the event is considered to be a Threshold Crossing event (Fig. A.11. In this case bit 0 indicates which threshold is exceeded and bit 1 shows, which ear is used for the AGC control.

A.2.6 jAER Control Panels and Biasgen Settings

The software control panels for the DASLP chip, the ADC and the AGC FPGA logic are presented in Figures A.12, A.13, A.14, A.15, A.16.

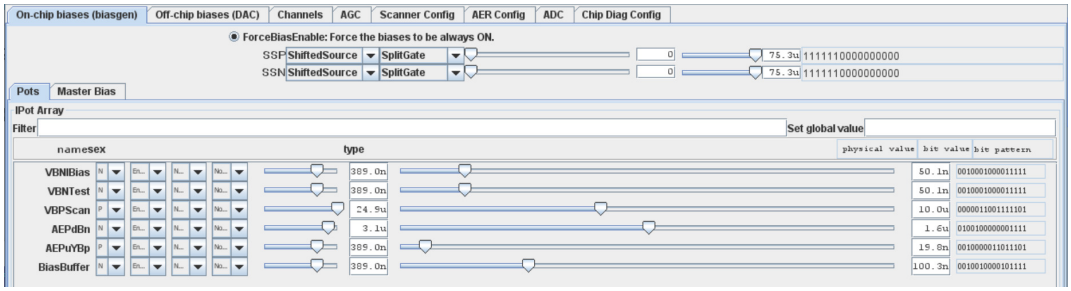

Figure A.12: jAER control panel for on-chip biases.

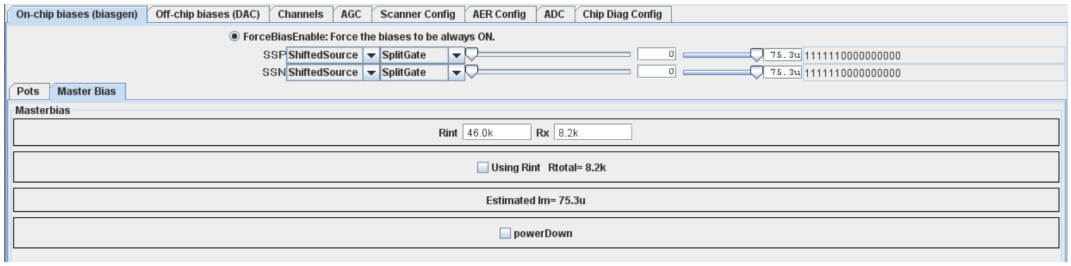

Figure A.13: jAER control panel for on-chip master bias.

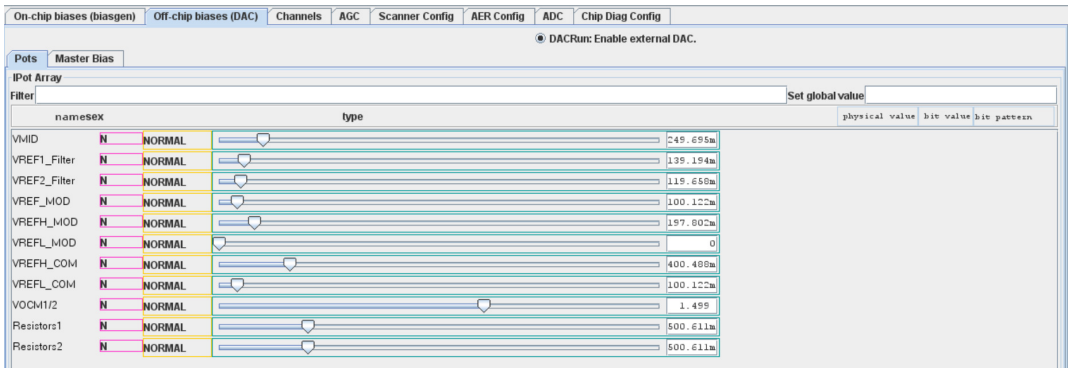

Figure A.14: jAER control panel for off-chip biases.

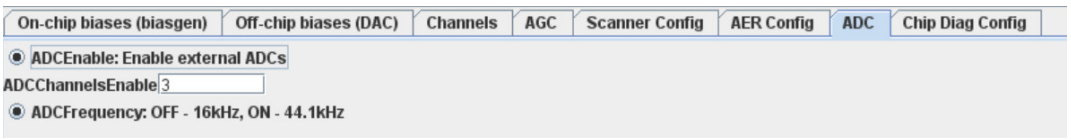

Figure A.15: jAER control panel for ADC. ADCChannelsEnable is a bitmask for enabling individual ADC channels: 0 – off, 1 – right, 2 – left, 3 – both.

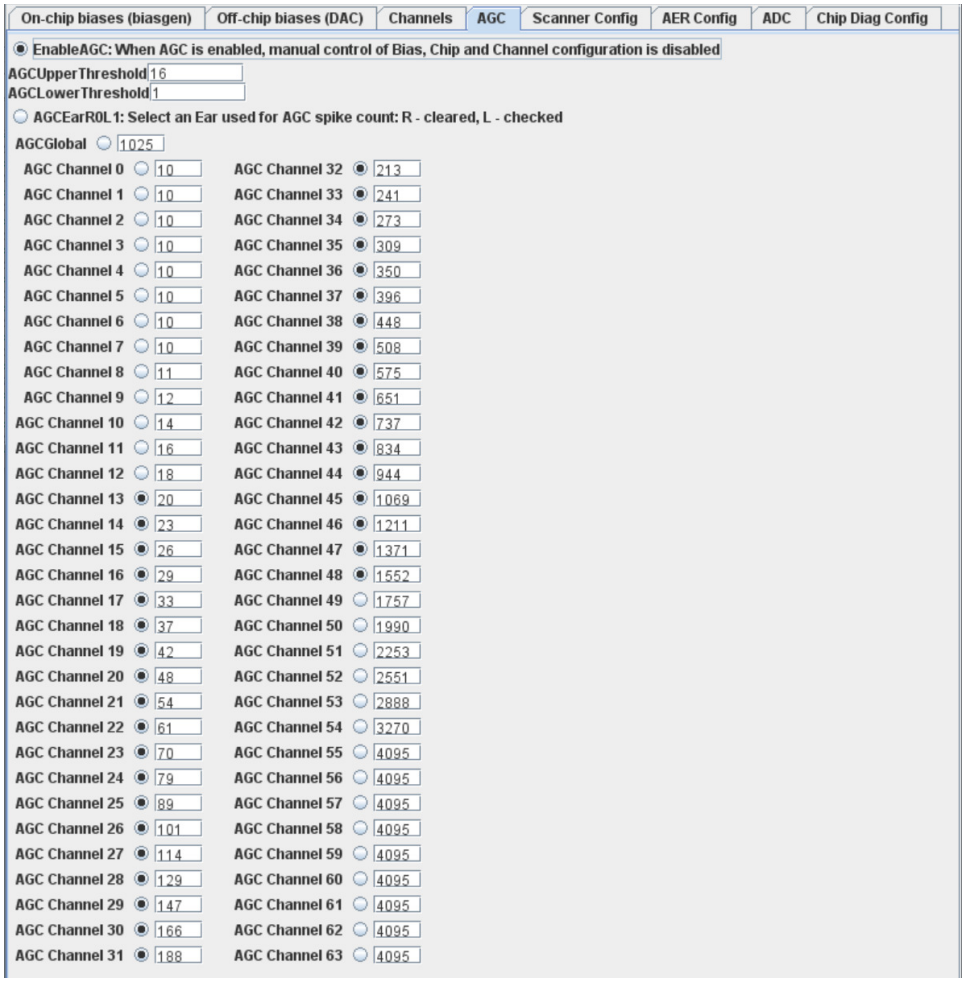

Figure A.16: jAER control panel for AGC. The upper and lower thresholds are given as a number of spikes within the averaging time interval. The averaging time interval is given in 0.1 ms units for each cochlea channel.

A.2.7 Measurements of Channel Characteristic Frequencies of DASLP

From the measurements of the frequency responses of the individual DASLP channels, we found that the characteristic frequencies of the channels cannot be fitted by a single exponential equation (thin black line on Fig. A.17). The base of the equation should be 1.13224, which would give a uniform channel spacing for 64 channels in the range 8 Hz to 20 kHz. Instead, we needed to fit the frequencies using two different exponential equations, one for the low frequency channels (blue circles in Fig. A.17), and one for high-frequency channels (blue crosses in Fig. A.17). We found that the high frequency channels (0 to 27) are spaced exponentially with a base equal to 1.12 and the low frequency channels (28 to 63) are spaced with the base equal to 1.141. Further we use these approximations to estimate the characteristic frequencies for all the channels that were not measured directly.

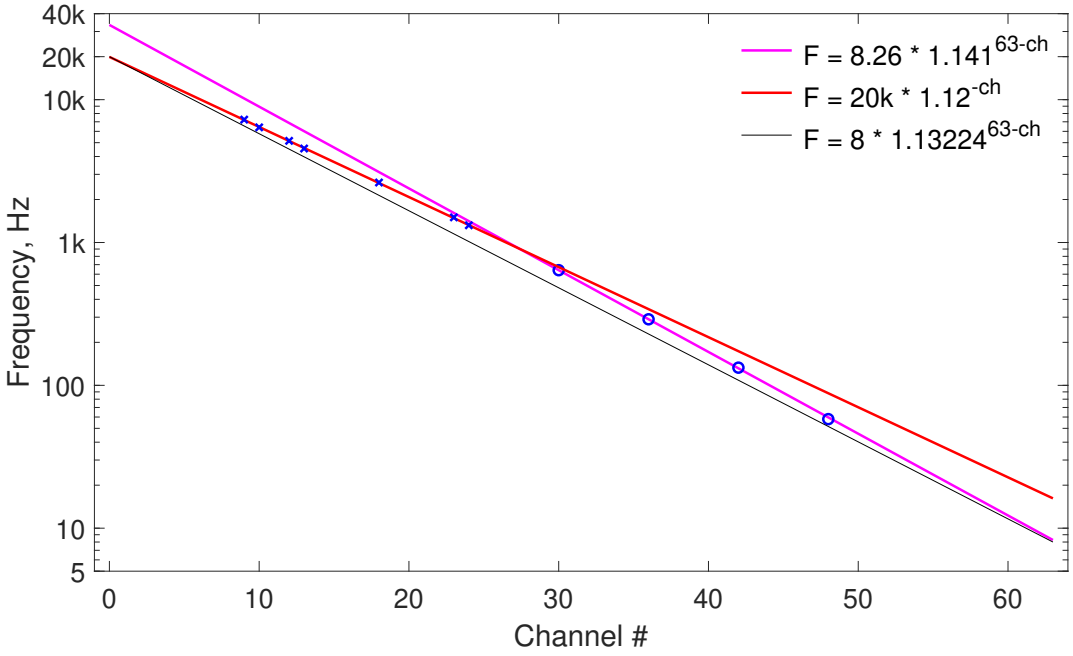

Figure A.17: Fitting measured characteristic frequencies of high- and low-frequency channels with two exponential functions. The best fit exponent for the high-frequency channels (blue crosses) has the base equal 1.12; and for the low-frequency channels (blue circles) the base is 1.141. The thin black line shows the characteristic frequencies distributed uniformly on a log scale between 8 and 20000 Hz.

A.2.8 Effect of Local Gain Control on Responses to a Two-Frequency Component Signal

We presume that AGC helps to preserve a shape of inter-spike intervals distribution of a channel in the presence of an interfering signal at the adjacent frequency bands. We studied how the shape of instantaneous frequencies histogram at channel 30 (480 Hz, 5 mV) changes in the presence of a 1 kHz interfering frequency at two signal-to-interference (SIR) levels: 0 and -6 dB. The results presented in Fig. A.18 show that there is a more prominent frequency peak that is closer to the target frequency in the AGC case for both SIR levels.

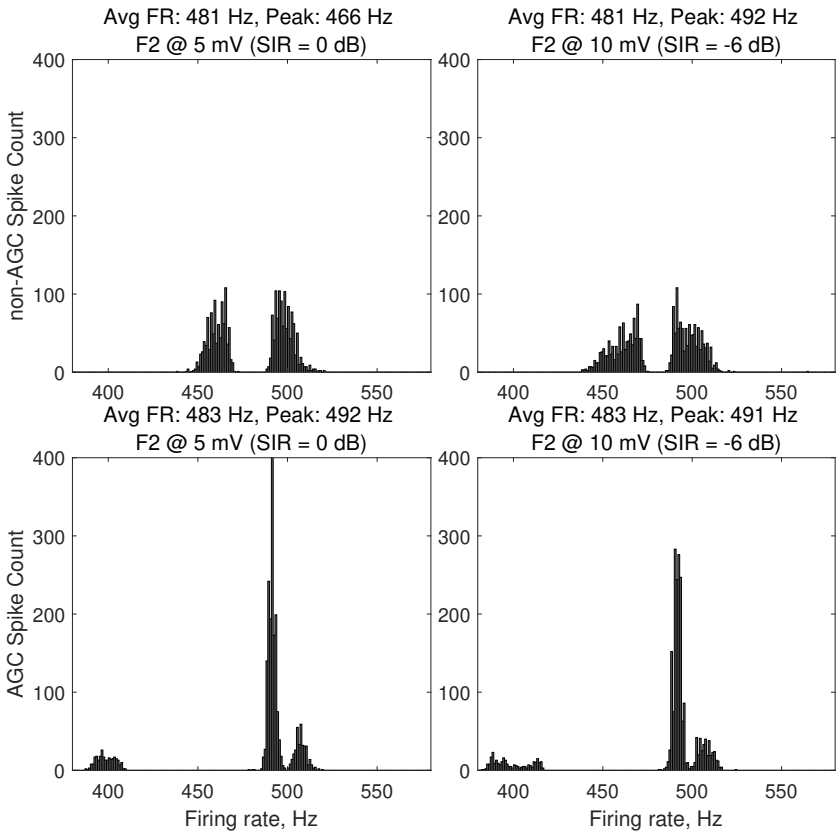

Figure A.18: Instantaneous firing rate (FR) of channel 30 (480 Hz) for non-AGC (top row) and AGC (bottom row) cases for a mixture of frequencies: F1 = 480 Hz @ 5 mV, F2 = 1 kHz @ 5, 10 mV.

A.3 SUPPLEMENTARY MATERIAL TO CHAPTER 4

A.3.1 WHISPER Platform Implementation Details

Figure A.19 shows the address space of the Wishbone FIFO used for data transmission from the FPGA to the Raspberry Pi board. The software running on the Raspberry Pi polls the status register at the FIFO_SZ+4 address and checks whether the Empty bit is cleared every 20 μs in order to provide low data transmission latency. In our implementation of the Wishbone interface the Empty flag is cleared only when the full FIFO buffer is filled with the data. This approach allows for reading the full buffer with one burst read command.

Address	READ	WRITE
[0:FIFO_SZ-1]	Data	Data
FIFO_SZ	RD_FIFO_SIZE	Reset FIFO and (Re)start acqusition
FIFO_SZ+1	WR_FIFO_SIZE	
FIFO_SZ+2	Read data count	
FIFO_SZ+3	Written data count	
FIFO_SZ+4	[15:4] 12-bit packet ID (counter) [3:0] [Empty, Stop, Stalled, Overflow]	

Figure A.19: Wishbone FIFO address space. FIFO_SZ is the FIFO size in 16-bit words. It was set to 256 for 8 kHz sampling rate and to 1024 for 24 kHz.

The format of the Wishbone read and write commands is shown in Fig. A.20. The 16-bit command (bits $31-16$) is followed by the 16-bit data payload if the Increment Address bit is set to 0. In order to initiate a multi-word data transaction (burst mode) the Increment Address bit has to be set to 1 and the required amount of data can be read or written without sending the address for each data word.

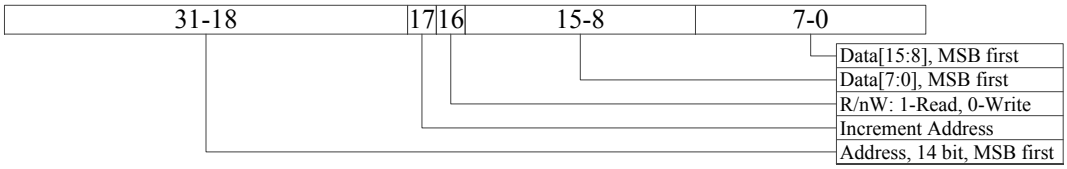

Figure A.20: Wishbone read/write commands format

Figure A.21 shows the PMOD connector pinout for two types of microphones: SPI and I²S. A front view of the 12-pin dual-row connector is depicted by a shadowed area. The logical bit numbers are given above and below the pin numbers. The SPI connection was used in the prototype version of the WHISPER platform. The specified SPI pinout allows direct attachment of the Digilent PMODMIC3[2] SPI digital microphones. The final WHISPER platform version uses custom designed I²S microphones, whose schematic is given in Fig. A.22. The designed PCB allows installing two models of microphones: ICS-43432[3] and ICS-43434[4] from InvenSense/TDK.

Bit #			3	2	1	0
PMOD Pin #	6	5	4	3	2	1
PMOD Pin #	12	11	10	9	8	7
Bit #			7	6	5	4
SPI Function	3.3V	GND	SCK	MISO	MOSI	SS
I2S Function	3.3V	GND	SD	SCK	WS	NC

Figure A.21: WHISPER PMOD connector pinout. The gray area represents the PMOD connector (front view).

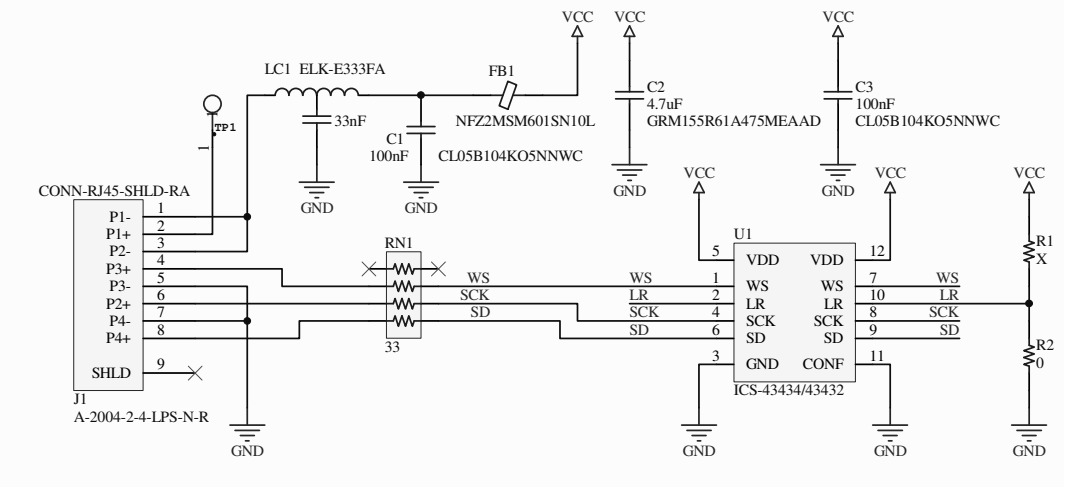

Figure A.22: Custom I²S microphone schematic diagram.

2 https://reference.digilentinc.com/_media/reference/pmod/pmodmic3/pmodmic3_rm.pdf
3 https://www.invensense.com/wp-content/uploads/2015/02/ICS-43432-data-sheet-v1.3.pdf
4 https://www.invensense.com/wp-content/uploads/2016/02/DS-000069-ICS-43434-v1.2.pdf

Figures A.23 and A.24 show a modification of the MRF89XAM9A module necessary for implementation of our synchronization algorithm. The MRF89XAM9A module schematic is adopted from the module datasheet[5].

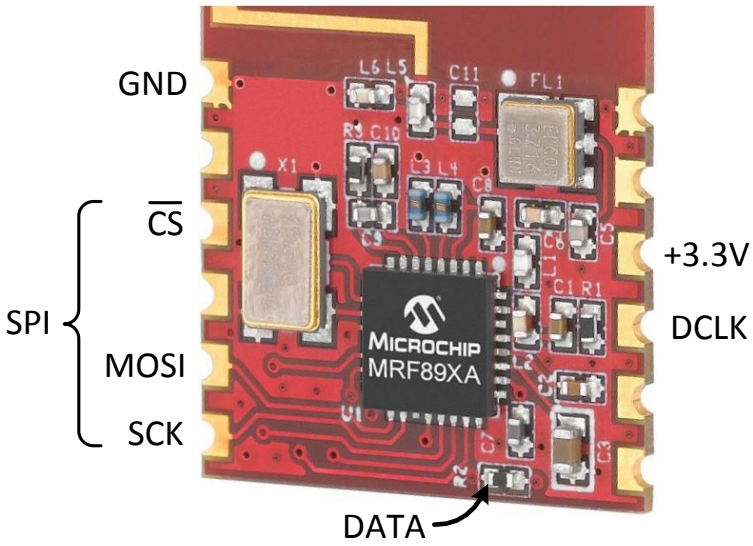

Figure A.23: MRF89XAM9A module pinout. DATA signal has no dedicated pin on the PCB, so a wire should be soldered to the R2 pad as indicated by the arrow.

5 https://ww1.microchip.com/downloads/en/DeviceDoc/75017B.pdf

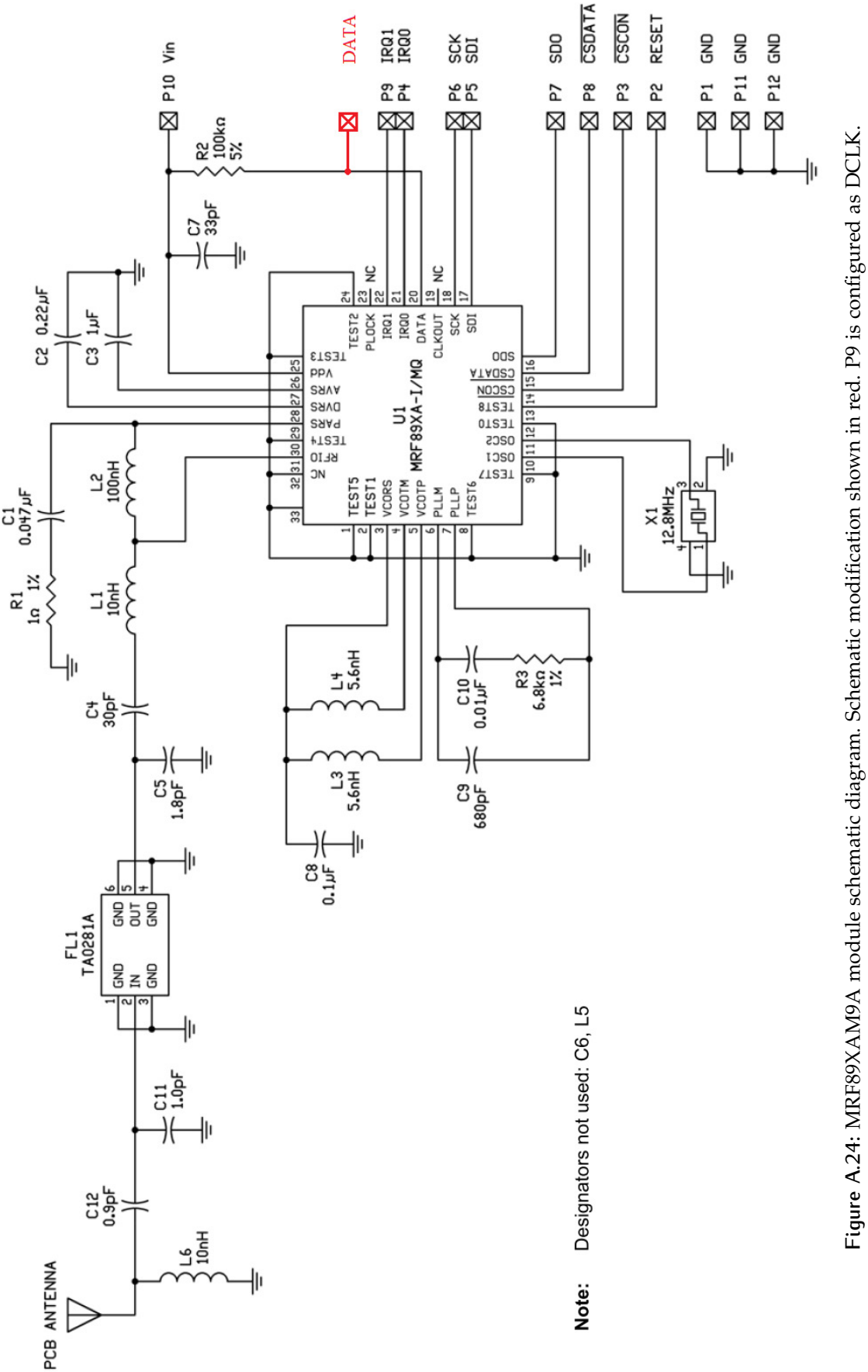

Figure A.24: MRF89XAM9A module schematic diagram. Schematic modification shown in red. P9 is configured as DCLK.

A.3.2 Phase Shift and Jitter of WHISPER Prototype

Phase synchronization measurements from two WHISPER-M4 modules are shown in Fig. A.25. The constant phase shift between the two modules depends on the relative spatial positions of the modules and is caused by the multipath radio wave propagation. The observed phase shift of 63 ns corresponds to a negligible spatial shift (much less than 1 mm) when sampling the acoustic signal in the air, so this value does not affect our processing algorithms. The second curve (green) shows the approximate cumulative distribution function of the sampling clock rising edge of the second module with respect to the first one. The measurements show an rms phase jitter (std dev) of about 100 ns.

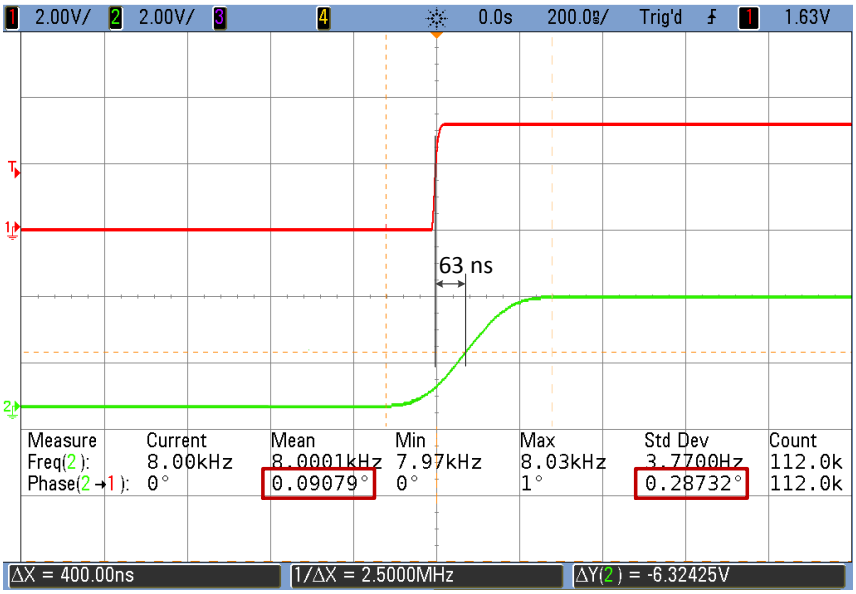

Figure A.25: Phase difference of the two wirelessly synchronized WHISPER-M4 prototype modules. The oscilloscope is synchronized on the front of the sampling clock of one of the modules (red). The sampling clock of the second module (green) is averaged over 112 thousand cycles. The mean shift of the second module clock w.r.t. the first one and the standard deviation of this shift are shown in the red boxes. The standard deviation corresponds to the RMS of sum of jitters of both modules. Assuming that the jitter is independent and has the same statistics at two modules, the jitter at one module is smaller than the measured value by a factor of $\sqrt{2}$.

A.3.3 Alternative Data Transmission Module

A photo of the DWM1000 UWB data transmission module soldered to a prototyping board with two PMOD connectors is shown in Fig. A.26.

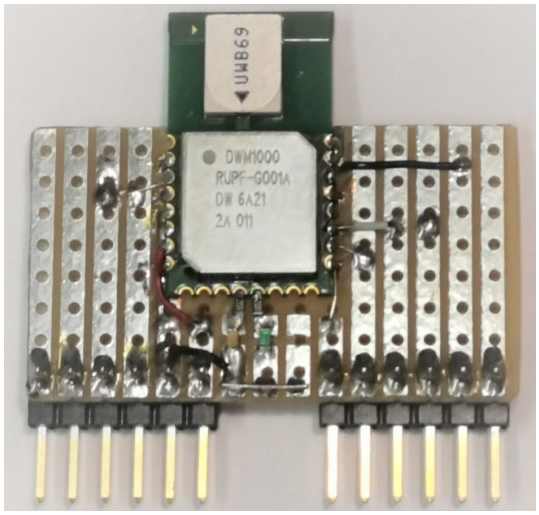

Figure A.26: DWM1000 wireless communication module.

Fig. A.27 shows a time-division multiplexing (TDM) scheme proposed in order to avoid a concurrent medium access, when several WHISPER-M4 modules are transmitting simultaneously, and hence, reduce the data transmission latency. An individual time slot is scheduled for each module, with short guard intervals between the time slots. Implementation of this scheme becomes possible when the clocks of all the WHISPER modules are synchronized with high precision. We achieved the clock synchronisation precision of about 200 ns, that allows to make the guard intervals less than 1 μs, which is negligibly small compared to the packet size (e.g. 602 μs for 512-byte packet, see Table 4.1 in Section 4.2.4). Two data transmission topologies are possible within this scheme: a full mesh, when at each time slot one module transmits and all other receive the signal; and a star topology, when a separate receiver (in the last row) collects the data from all the modules.

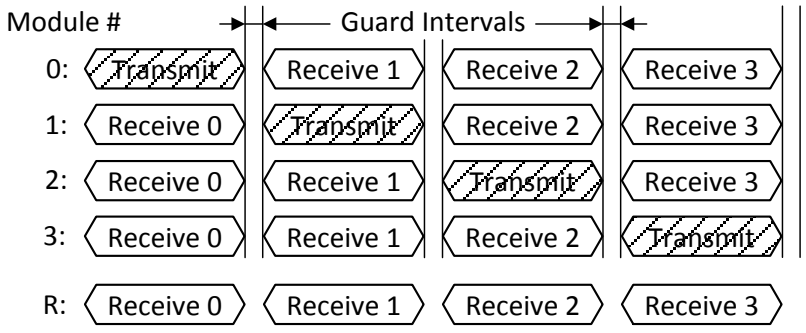

Figure A.27: Time-division multiplexing scheme to be used with the DWM1000 transceiver. Rows $0-3$ represent WHISPER-M4 modules, and the last row represents a separate receiver module.

ACRONYMS

ADC	Analog to Digital Converter
AER	Address Event Representation
AFE	Acoustic Feature Extraction
AGC	Automatic Gain Control
ANN	Artificial Neural Network
ASIC	Application-Specific Integrated Circuit
BSS	blind source separation
CNN	Convolutional Neural Network
COTS	commercially available off-the-shelf
DAS	Dynamic Audio Sensor
DASLP	Dynamic Audio Sensor Low Power
DBN	Deep Belief Network
DNN	Deep Neural Network
DVS	Dynamic Vision Sensor
EEG	electroencephalogram
FIFO	First In, First Out
FIR	Finite Impulse Response
FLL	Frequency Locked Loop
FPGA	Field-Programmable Gate Array
FM	Frequency Modulation
FSK	frequency-shift keying
FSM	Finite State Machine
GPS	Global Positioning System
I^2S	Inter-IC Sound
IFR	instantaneous firing rate
IIR	Infinite Impulse Response
IO	Input-Output
ISI	interspike interval
ISM	industrial, scientific and medical
JND	just noticeable difference
KWS	Keyword Spotting
LCMV	Linearly Constrained Minimum Variance

LIF	Leaky Integrate-and-Fire
LR	Logistic Regression
MASS	Multi-channel Audio Source Separation
MNIST	Modified National Institute of Standards and Technology
NTP	Network Time Protocol
OS	Operating System
PC	Personal Computer
PCB	Printed Circuit Board
PGA	Programmable Gain Amplifier
PLL	Phase-locked Loop
PMOD	Peripheral Module
PSS	Point of Subjective Simultaneity
RBM	Restricted Boltzmann Machines
RF	Radio Frequency
RMS	Root Mean Square
SBC	single board computer
SIR	Signal-to-Interference Ratio
SNN	Spiking Neural Network
SNR	signal-to-noise ratio
SoC	System on a Chip
SPI	Serial Peripheral Interface
SPL	Sound Pressure Level
STD	Standard Deviation
STOI	Short-Time Objective Intelligibility
TDM	time-division multiplexing
UDP	User Datagram Protocol
UWB	Ultra-Wideband
VAD	Voice Activity Detection
WASN	Wireless Acoustic Sensor Network

PUBLICATIONS RESULTING FROM THIS WORK

[1] E. Ceolini, J. Anumula, A. E. Huber, I. Kiselev, and S.-C. Liu, "Speaker activity detection and minimum variance beamforming for source separation.," in *Interspeech*, 2018, pp. 836–840.

[2] E. Ceolini, I. Kiselev, and S.-C. Liu, "Audio classification systems using deep neural networks and an event-driven auditory sensor," *2019 IEEE SENSORS*, 2019.

[3] E. Ceolini, I. Kiselev, and S.-C. Liu, "Evaluating multi-channel multi-device speech separation algorithms in the wild: A hardware-software solution," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 1428–1439, 2020.

[4] C. Gao, S. Braun, I. Kiselev, J. Anumula, T. Delbruck, and S.-C. Liu, "Real-time speech recognition for IoT purpose using a delta recurrent neural network accelerator," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2019, pp. 1–5.

[5] I. Kiselev, E. Ceolini, D. Wong, A. d. Cheveigne, and S.-C. Liu, "WHISPER: Wirelessly synchronized distributed audio sensor platform," in *2017 IEEE 42nd Conference on Local Computer Networks Workshops (LCN Workshops)*, Oct. 2017, pp. 35–43.

[6] I. Kiselev and S.-C. Liu, "Event-driven local gain control on a spiking cochlea sensor," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021. DOI: 10.1109/ISCAS51556.2021.9401742.

[7] I. Kiselev, D. Neil, and S.-C. Liu, "Event-driven deep neural network hardware system for sensor fusion," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2016, pp. 2495–2498.

[8] I. Kiselev, D. Neil, and S.-C. Liu, "Live demonstration: Event-driven deep neural network hardware system for sensor fusion," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2016, pp. 452–452.

BIBLIOGRAPHY

[1] *3D Audiosense*, http://www.3daudiosense.com/uploads/2/4/7/2/24724752/zylia-audiosense-parp_polish_product_of_the_future.pdf, 2014.

[2] H. Abdalla and T. K. Horiuchi, "An ultrasonic filterbank with spiking neurons," in *2005 IEEE International Symposium on Circuits and Systems*, 2005, pp. 4201–4204.

[3] J. Acharya, A. Patil, X. Li, Y. Chen, S.-C. Liu, and A. Basu, "A comparison of low-complexity real-time feature extraction for neuromorphic speech recognition," *Frontiers in Neuroscience*, vol. 12, p. 160, 2018.

[4] H. Afifi, J. Schmalenstroeer, J. Ullmann, R. Häb-Umbach, and H. Karl, "Marvelo: A framework for signal processing in wireless acoustic sensor networks," in *Speech Communication; 13th ITG-Symposium*, VDE, 2018, pp. 1–5.

[5] J. Anumula, D. Neil, X.-Y. Li, T. Delbruck, and S.-C. Liu, "Live demonstration: Event-driven real-time spoken digit recognition system," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2017.

[6] J. Anumula, E. Ceolini, Z. He, A. Huber, and S.-C. Liu, "An event-driven probabilistic model of sound source localization using cochlea spikes," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018, pp. 1–5.

[7] J. Anumula, D. Neil, T. Delbruck, and S.-C. Liu, "Feature representations for neuromorphic audio spike streams," *Frontiers in Neuroscience*, vol. 12, 2018. DOI: 10.3389/fnins.2018.00023. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2018.00023.

[8] J. Benesty, J. Chen, Y. Huang, and J. Dmochowski, "On microphone-array beamforming from a mimo acoustic signal processing perspective," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 3, pp. 1053–1065, 2007.

[9] A. Bertrand, "Applications and trends in wireless acoustic sensor networks: A signal processing perspective," in *2011 18th IEEE symposium on communications and vehicular technology in the Benelux (SCVT)*, 2011, pp. 1–6.

[10] A. Bertrand, S. Doclo, S. Gannot, N. Ono, and T. van Waterschoot, "Special issue on wireless acoustic sensor networks and ad hoc microphone arrays," *Signal Processing*, vol. 107, no. C, pp. 1–3, 2015.

[11] E. Ceolini, J. Anumula, S. Braun, and S.-C. Liu, "Event-driven pipeline for low-latency low-compute keyword spotting and speaker verification system," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, pp. 7953–7957. DOI: 10.1109/ICASSP.2019.8683669.

[12] E. Ceolini, I. Kiselev, and S.-C. Liu, "Audio classification systems using deep neural networks and an event-driven auditory sensor," in *2019 IEEE Sensors*, 2019, pp. 1–4.

[13] E. Ceolini, I. Kiselev, and S.-C. Liu, "Evaluating multi-channel multi-device speech separation algorithms in the wild: A hardware-software solution," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 1428–1439, 2020.

[14] E. Ceolini, D. Neil, T. Delbruck, and S.-C. Liu, "Temporal sequence recognition in a self-organizing recurrent network," in *2016 Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, 2016, pp. 1–4.

[15] V. Chan, S. C. Liu, and A. van Schaik, "AER EAR: A matched silicon cochlea pair with address event representation interface," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 1, pp. 48–59, Jan. 2007, ISSN: 1549-8328. DOI: 10.1109/TCSI.2006.887979.

[16] V. Y.-S. Chan, C. T. Jin, and A. van Schaik, "Neuromorphic audio-visual sensor fusion on a sound-localising robot," *Frontiers in neuroscience*, vol. 6, p. 21, 2012.

[17] J. Coady, D. Toal, T. Newe, and G. Dooly, "Remote acoustic analysis for tool condition monitoring," *Procedia Manufacturing*, vol. 38, pp. 840–847, 2019, 29th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM 2019), June 24-28, 2019, Limerick, Ireland, Beyond Industry 4.0: Industrial Advances, Engineering Education and Intelligent Manufacturing, ISSN: 2351-9789. DOI: https://doi.org/10.1016/j.promfg.2020.01.165. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2351978920301669.

[18] P. U. Diehl, D. Neil, J. Binas, M. Cook, S. C. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *Proceedings of the International Joint Conference on Neural Networks*, Killarney, Ireland, 2015, ISBN: 9781479919604. DOI: 10.1109/IJCNN.2015.7280696.

[19] H. Finger and S.-C. Liu, "Estimating the location of a sound source with a spike-timing localization algorithm," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2011, pp. 2461–2464.

[20] O. L. Frost, "An algorithm for linearly constrained adaptive array processing," *Proceedings of the IEEE*, vol. 60, no. 8, pp. 926–935, 1972.

[21] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The spinnaker project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, May 2014, ISSN: 0018-9219. DOI: 10.1109/JPROC.2014.2304638.

[22] C. Gao, S. Braun, I. Kiselev, J. Anumula, T. Delbruck, and S. Liu, "Real-time speech recognition for IoT purpose using a delta recurrent neural network accelerator," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2019, pp. 1–5. DOI: 10.1109/ISCAS.2019.8702290.

[23] C. Gao, A. Rios-Navarro, X. Chen, T. Delbruck, and S.-C. Liu, "EdgeDRNN: Enabling low-latency recurrent neural network edge inference," in *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2020, pp. 41–45.

[24] J. Garofolo, L. Lamel, W. Fisher, J. Fiscus, D. Pallett, N. Dahlgren, and V. Zue, "Timit acoustic-phonetic continuous speech corpus," *Linguistic Data Consortium*, Nov. 1992.

[25] L. Girod, M. Lukac, V. Trifa, and D. Estrin, "The design and implementation of a self-calibrating distributed acoustic sensing platform," in *Proceedings of the 4th international conference on Embedded networked sensor systems*, Jan. 2006, pp. 71–84. DOI: 10.1145/1182807.1182815.

[26] N. Goux, J.-B. Casanova, G. Pillonnet, and F. Badets, "A 6nw 0.0013mm2 ilo bandpass filter for time-based feature extraction," *IEEE Solid-State Circuits Letters*, vol. PP, pp. 1–1, Aug. 2020. DOI: 10.1109/LSSC.2020.3016716.

[27] T. J. Hamilton, C. Jin, A. Van Schaik, and J. Tapson, "An active 2-d silicon cochlea," *IEEE Transactions on biomedical circuits and systems*, vol. 2, no. 1, pp. 30–43, 2008.

[28] J. R. Higgins, *Sampling theory in Fourier and signal analysis: foundations*. Oxford University Press on Demand, 1996.

[29] I. J. Hirsh and J. Sherrick Carl E, "Perceived order in different sense modalities.," *Journal of experimental psychology*, vol. 62, no. 5, 1961, ISSN: 0022-1015.

[30] A. E. G. Huber and S. Liu, "On send-on-delta sampling of bandlimited functions," in *2017 International Conference on Sampling Theory and Applications (SampTA)*, IEEE, 2017, pp. 422–426. DOI: 10.1109/SAMPTA.2017.8024346.

[31] A. E. G. Huber and S. Liu, "On approximation of bandlimited functions with compressed sensing," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 4009–4013. DOI: 10.1109/ICASSP.2018.8461376.

[32] A. Ipser, V. Agolli, A. Bajraktari, F. Al-Alawi, N. Djaafara, and E. Freeman, "Sight and sound persistently out of synch: Stable individual differences in audiovisual synchronisation revealed by implicit measures of lip-voice integration," *Scientific Reports*, vol. 7, May 2017. DOI: 10.1038/srep46413.

[33] ITU-R Recommendation, "Effects of building materials and structures on radiowave propagation above about 100 MHz," Tech. Rep., Jul. 2015.

[34] *jAER project*, http://jaerproject.org, 2007.

[35] A. G. Katsiamis, E. M. Drakakis, and R. F. Lyon, "A biomimetic, 4.5 μW, 120+ dB, log-domain cochlea channel with AGC," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 3, pp. 1006–1022, Mar. 2009, ISSN: 0018-9200. DOI: 10.1109/JSSC.2008.2011039.

[36] T. Kawase, I. Yahata, A. Kanno, S. Sakamoto, Y. Takanashi, S. Takata, N. Nakasato, R. Kawashima, and Y. Katori, "Impact of audio-visual asynchrony on lip-reading effects -neuromagnetic and psychophysical study-," *PLOS ONE*, vol. 11, e0168740, Dec. 2016. DOI: 10.1371/journal.pone.0168740.

[37] M. Keetels and J. Vroomen, "Perception of synchrony between the senses," in *The neural bases of multisensory processes*, CRC Press/Taylor & Francis, 2012.

[38] I. Kiselev, E. Ceolini, D. Wong, A. d. Cheveigne, and S.-C. Liu, "WHISPER: Wirelessly synchronized distributed audio sensor platform," in *2017 IEEE 42nd Conference on Local Computer Networks Workshops (LCN Workshops)*, Oct. 2017, pp. 35–43.

[39] I. Kiselev, E. Ceolini, D. Wong, A. d. Cheveigne, and S.-C. Liu, "WHISPER: Wirelessly synchronized distributed audio sensor platform," in *2017 IEEE 42nd Conference on Local Computer Networks Workshops (LCN Workshops)*, Oct. 2017, pp. 35–43.

[40] I. Kiselev and S.-C. Liu, "Event-driven local gain control on a spiking cochlea sensor," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021. DOI: 10.1109/ISCAS51556.2021.9401742.

[41] I. Kiselev, D. Neil, and S.-C. Liu, "Event-driven deep neural network hardware system for sensor fusion," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2016, pp. 2495–2498.

[42] I. Kiselev, D. Neil, and S.-C. Liu, "Live demonstration: Event-driven deep neural network hardware system for sensor fusion," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2016, pp. 452–452.

[43] P. Klein, J. Conradt, and S.-C. Liu, "Scene stitching with event-driven sensors on a robot head platform," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2015, pp. 2421–2424.

[44] D. Y. Levin, E. A. Habets, and S. Gannot, "On the average directivity factor attainable with a beamformer incorporating null constraints," *IEEE Signal Processing Letters*, vol. 22, no. 11, pp. 2122–2126, 2015.

[45] C.-H. Li, T. Delbrück, and S.-C. Liu, "Real-time speaker identification using the AEREAR2 event-based silicon cochlea," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2012, pp. 1159–1162.

[46] C. Li, C. Brandli, R. Berner, H. Liu, M. Yang, S.-C. Liu, and T. Delbruck, "Design of an RGBW color VGA rolling and global shutter dynamic and active-pixel vision sensor," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, 2015, pp. 718–721.

[47] X. Li, D. Neil, T. Delbruck, and S.-C. Liu, "Lip reading deep network exploiting multi-modal spiking visual and auditory sensors," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019, pp. 1–5.

[48] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 $\times$ 128 120 dB 15 μs latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Feb. 2008, ISSN: 0018-9200.

[49] R. Lienhart, I. Kozintsev, S. Wehr, and M. Yeung, "On the importance of exact synchronization for distributed audio signal processing," in *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, IEEE, vol. 4, 2003, pp. IV–840.

[50] S.-C. Liu, A. van Schaik, B. Minch, and T. Delbrück, "Asynchronous binaural spatial audition sensor with 2 × 64 × 4 channel output," *IEEE Trans. Biomed. Circuits Syst.*, vol. 8, no. 4, pp. 453–464, 2014. DOI: 10.1109/TBCAS.2013.2281834.

[51] S.-C. Liu, T. Delbruck, G. Indiveri, R. Douglas, and A. Whatley, *Event-Based Neuromorphic Systems*. Chichester, UK: John Wiley & Sons, 2015, p. 440, ISBN: 0470018496. [Online]. Available: https://books.google.com/books?id=MuvsBQAAQBAJ&pgis=1.

[52] S.-C. Liu and R. Douglas, "Temporal coding in a silicon network of integrate-and-fire neurons," *IEEE Transactions on Neural Networks*, vol. 15, no. 5, pp. 1305–1314, 2004.

[53] *Lockit*, http://ambient.de/en/product_custom_cat/Timecode-en, 2019.

[54] R. F. Lyon, "Cascades of two-pole–two-zero asymmetric resonators are good models of peripheral auditory function," *The Journal of the Acoustical Society of America*, vol. 130, no. 6, pp. 3893–3904, 2011.

[55] R. F. Lyon, A. G. Katsiamis, and E. M. Drakakis, "History and future of auditory filter models," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, 2010, pp. 3809–3812.

[56] A. Maat, L. Trost, H. Sagunsky, S. Seltmann, and M. Gahr, "Zebra finch mates use their forebrain song system in unlearned call communication," *PloS one*, vol. 9, e109334, Oct. 2014. DOI: 10.1371/journal.pone.0109334.

[57] S. Markovich-Golan, S. Gannot, and I. Cohen, "Distributed multiple constraints generalized sidelobe canceler for fully connected wireless acoustic sensor networks," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 2, pp. 343–356, 2012.

[58] S. Miyabe, N. Ono, and S. Makino, "Blind compensation of interchannel sampling frequency mismatch for ad hoc microphone array based on maximum likelihood estimation," *Signal Processing*, vol. 107, pp. 185–196, 2015.

[59] D. P. Moeys, T. Delbrück, and S.-C. Liu, "Current-mode automated quality control cochlear resonator for bird identity tagging," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2015, pp. 1734–1737.

[60] D. Neil and S.-C. Liu, "Minitaur, an event-driven FPGA-based spiking network accelerator," *IEEE Trans on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 12, pp. 2621–2628, 2014.

[61] D. Neil and S. C. Liu, "Effective sensor fusion with event-based sensors and deep network architectures," *Proceedings - IEEE International Symposium on Circuits and Systems*, vol. 2016-July, pp. 2282–2285, 2016, ISSN: 02714310. DOI: 10.1109/ISCAS.2016.7539039.

[62] P. O'Connor, D. Neil, S.-C. Liu, T. Delbruck, and M. Pfeiffer, "Real-time classification and sensor fusion with a spiking Deep Belief Network," *Frontiers in Neuroscience*, vol. 7, 2013.

[63] K. Ochi, S. Miyabe, and S. Makino, "Multi-talker speech recognition based on blind source separation with ad-hoc microphone array using smartphones and cloud storage.," in *Interspeech*, 2016, pp. 3369–3373.

[64] S. Oh, M. Cho, Z. Shi, J. Lim, Y. Kim, S. Jeong, Y. Chen, R. Rothe, D. Blaauw, H.-S. Kim, and D. Sylvester, "An acoustic signal processing chip with 142-nw voice activity detection using mixer-based sequential frequency scanning and neural network classification," *IEEE Journal of Solid-State Circuits*, vol. PP, pp. 1–12, Sep. 2019. DOI: 10.1109/JSSC.2019.2936756.

[65] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, and T. Delbruck, "Retinomorphic event-based vision sensors: Bioinspired cameras with spiking output," *Proceedings of the IEEE*, vol. 102, no. 10, pp. 1470–1484, Oct. 2014, ISSN: 0018-9219. DOI: 10.1109/JPROC.2014.2346153.

[66] C. K. Reddy, E. Beyrami, J. Pool, R. Cutler, S. Srinivasan, and J. Gehrke, "A scalable noisy speech dataset and online subjective test framework," *Proc. Interspeech 2019*, pp. 1816–1820, 2019.

[67] B. Rueckauer and S. C. Liu, "Conversion of analog to spiking neural networks using sparse temporal coding," in *Proceedings - IEEE International Symposium on Circuits and Systems*, Florence, Italy: IEEE, 2018, pp. 8–12, ISBN: 9781538648810. DOI: 10.1109/ISCAS.2018.8351295.

[68] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, "Conversion of Continuous-Valued Deep Networks to Efficient Event-Driven Networks for Image Classification," *Frontiers in Neuroscience*, vol. 11, no. December, pp. 1–12, 2017, ISSN: 1662-453X. DOI: 10.3389/fnins.2017.00682. [Online]. Available: http://journal.frontiersin.org/article/10.3389/fnins.2017.00682/full.

[69] M. A. Ruggero, "Responses to sound of the basilar membrane of the mammalian cochlea," *Current Opinion in Neurobiology*, vol. 2, no. 4, pp. 449–456, 1992.

[70] E. Sazonov, V. Krishnamurthy, and R. Schilling, "Wireless intelligent sensor and actuator network - a scalable platform for time-synchronous applications of structural health monitoring," *Structural Health Monitoring*, vol. 9, no. 5, pp. 465–476, 2010. DOI: 10.1177/1475921710370003. [Online]. Available: https://doi.org/10.1177/1475921710370003.

[71] J. Schmalenstroeer, P. Jebramcik, and R. Häb-Umbach, "A combined hardware-software approach for acoustic sensor network synchronization," *Signal Process.*, vol. 107, no. C, pp. 171–184, Feb. 2015, ISSN: 0165-1684.

[72] C. Schörkhuber, M. Zaunschirm, and I. Zmölnig, "Wilma-wireless largescale microphone array," in *Linux Audio Conference*, vol. 2014, 2014.

[73] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gomez-Rodriguez, L. Camunas-Mesa, R. Berner, M. Rivas, T. Delbruck, S.-C. Liu, R. Douglas, P. Häfliger, G. Jimenez-Moreno, A. Civit, T. Serrano-Gotarredona, A. Acosta-Jimenez, and B. Linares-Barranco, "CAVIAR: A 45K-neuron, 5M-synapse, 12G-connects/sec AER hardware sensory-

processing-learning-actuating system for high speed visual object recognition and tracking," *IEEE Trans. Neural Netw.*, vol. 20, no. 9, pp. 1417–1438, 2009.

[74] D. Snyder, G. Chen, and D. Povey, *MUSAN: A Music, Speech, and Noise Corpus*, arXiv:1510.08484v1, 2015. eprint: 1510.08484.

[75] E. Stromatias, D. Neil, F. Galluppi, M. Pfeiffer, S.-C. Liu, and S. Furber, "Live demonstration: Handwritten digit recognition using spiking Deep Belief Networks on SpiNNaker," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2015, pp. 1901–1901.

[76] E. Stromatias, D. Neil, M. Pfeiffer, F. Galluppi, S. B. Furber, and S.-C. Liu, "Robustness of spiking Deep Belief Networks to noise and reduced bit precision of neuro-inspired hardware platforms," *Frontiers in Neuroscience*, vol. 9, 2015.

[77] W. Y. Tsai, D. Barch, A. Cassidy, M. Debole, A. Andreopoulos, B. Jackson, M. Flickner, J. Arthur, D. Modha, J. Sampson, and V. Narayanan, "Always-on speech recognition using TrueNorth, a reconfigurable, neurosynaptic processor," *IEEE Transactions on Computers*, vol. PP, no. 99, pp. 1–1, 2016, ISSN: 0018-9340. DOI: 10.1109/TC.2016.2630683.

[78] I. Uysal, H. Sathyendra, and J. G. Harris, "Spike-based feature extraction for noise robust speech recognition using phase synchrony coding," in *2007 IEEE International Symposium on Circuits and Systems*, 2007, pp. 1529–1532. DOI: 10.1109/ISCAS.2007.378702.

[79] I. Uysal, H. Sathyendra, and J. Harris, "Towards spike-based speech processing: A biologically plausible approach to simple acoustic classification," *International Journal of Applied Mathematics and Computer Science*, vol. 18, no. 2, pp. 129–137, 2008.

[80] A. van Schaik, V. Chan, and C. Jin, "Sound localisation with a silicon cochlea pair," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009, pp. 2197–2200. DOI: 10.1109/ICASSP.2009.4960054.

[81] B. D. Van Veen, W. Van Drongelen, M. Yuchtman, and A. Suzuki, "Localization of brain electrical activity via linearly constrained minimum variance spatial filtering," *IEEE Transactions on Biomedical Engineering*, vol. 44, no. 9, pp. 867–880, 1997.

[82] V. Van Wassenhove, K. Grant, and D. Poeppel, "Temporal window of integration in auditory–visual speech perception," *Neuropsychologia*, vol. 45, pp. 598–607, Mar. 2007. DOI: 10.1016/j.neuropsychologia.2006.01.001.

[83] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE transactions on audio, speech, and language processing*, vol. 14, no. 4, pp. 1462–1469, 2006.

[84] J. Vroomen and J. Stekelenburg, "Perception of intersensory synchrony in audiovisual speech: Not that special," *Cognition*, vol. 118, pp. 75–83, Oct. 2010. DOI: 10.1016/j.cognition.2010.10.002.

[85] J. van der Vyver, A. Kern, and R. Stoop, "Active part implementation of a biomorphic Hopf cochlea," in *Proceedings of the European Conference on Circuit Theory and Design ECCTD 2003*, 2003, pp. 285–288.

[86] B. Wen and K. Boahen, "A silicon cochlea with active coupling," *IEEE Trans. Biomed. Circuits Syst.*, vol. 3, no. 6, pp. 444–455, 2009.

[87] A. Winkler, M. Latzel, and I. Holube, "Open versus closed hearing-aid fittings: A literature review of both fitting approaches," *Trends in hearing*, vol. 20, p. 2 331 216 516 631 741, 2016.

[88] J. Wu, Y. Chua, M. Zhang, H. Li, and K. C. Tan, "A spiking neural network framework for robust sound classification," *Frontiers in Neuroscience*, vol. 12, p. 836, 2018.

[89] Y. Xu, C. S. Thakur, R. K. Singh, T. J. Hamilton, R. M. Wang, and A. van Schaik, "A fpga implementation of the car-fac cochlear model," *Frontiers in neuroscience*, vol. 12, p. 198, 2018.

[90] G. Yang, R. F. Lyon, and E. M. Drakakis, "A 6 μW per channel analog biomimetic cochlear implant processor filterbank architecture with across channels agc," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 9, no. 1, pp. 72–86, 2015.

[91] M. Yang, C. H. Chien, T. Delbruck, and S. C. Liu, "A 0.5 V 55 μW 64 $\times$ 2 channel binaural silicon cochlea for event-driven stereo-audio sensing," *IEEE Journal of Solid-State Circuits*, vol. 51, no. 11, pp. 2554–2569, Nov. 2016, ISSN: 0018-9200. DOI: 10.1109/JSSC.2016.2604285.

[92] M. Yang, C. Yeh, Y. Zhou, J. P. Cerqueira, A. A. Lazar, and M. Seok, "A 1 μW voice activity detector using analog feature extraction and digital deep neural network," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, Feb. 2018, pp. 346–348. DOI: 10.1109/ISSCC.2018.8310326.

[93] M. Yang, C. Yeh, Y. Zhou, J. P. Cerqueira, A. A. Lazar, and M. Seok, "Design of an always-on deep neural network-based 1 μW voice activity detector aided with a customized software model for analog feature extraction," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 6, pp. 1764–1777, 2019. DOI: 10.1109/JSSC.2019.2894360.

[94] A. T. Zai, S. Bhargava, N. Mesgarani, and S.-C. Liu, "Reconstruction of audio waveforms from spike trains of artificial cochlea models," *Frontiers in Neuroscience*, vol. 9, p. 347, 2015.

[95] C. Zamarreno-Ramos, A. Linares-Barranco, T. Serrano-Gotarredona, and B. Linares-Barranco, "Multicasting mesh AER: A scalable assembly approach for reconfigurable neuromorphic structured AER systems. Application to ConvNets," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 7, no. 1, pp. 82–102, Feb. 2013, ISSN: 1932-4545. DOI: 10.1109/TBCAS.2012.2195725.

[96] M. Zohourian and R. Martin, "Binaural speaker localization and separation based on a joint ITD/ILD model and head movement tracking," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 430–434.