



POLITECNICO DI TORINO

Master's Degree in Biomedical Engineering

Master's Degree Thesis

FPGA-Based Event-Driven Neural Spike Acquisition System for 60-Channel Microelectrode Arrays for In Vitro Cultures

Supervisor

Prof. Eng. Danilo DEMARCHI

Co-supervisors:

Ph.D. Alessandro SANGINARIO

Ph.D Candidate Marco BOSCHERINI

Ph.D Candidate Luigi GIUFFRIDA

Candidate

Mattia LAI

matricola: 333321

ACADEMIC YEAR 2025-2026

Abstract

The development of microelectrode array (MEA) technology, interfaced with low-noise, high-gain analog conditioning circuits, has become a pillar for evaluating the electrophysiological properties of neurons in vitro. Simultaneous, parallel access to the activity of large neural populations within a network is essential for investigating information processing, learning, and the emergence of collective behavior. The unique ability of in vitro cultures to self-organise into functional synaptic networks over weeks, combined with MEA-based recording, has opened pathways toward active, bidirectional bio-hybrid interfaces, with demonstrations ranging from pharmacological screening to goal-directed behaviour in simulated environments.

Scaling these systems to many recording channels, however, introduces compounding challenges at every layer. In fact, conventional MEA acquisition platforms continuously digitize the full waveform from every electrode at sampling rates of 10–50 kS/s, generating data volumes that increase with both channel count and session duration. As a result, these devices become difficult to manage in terms of both data storage and subsequent analysis. Moreover, commercial systems capable of handling such throughput can cost up to \$100,000 for high-density configurations, placing them out of reach for many research groups.

For a broad class of network-level experiments such as burst detection, synchrony quantification, firing rate estimation, and real-time closed-loop stimulation, standard devices that analyse the waveform morphology of individual spikes are not necessary. In contrast, what really matters is the spatiotemporal activation map, which means that the primary information is simply whether a neuron fired and when.

This thesis, which is part of a broader project aiming to develop a low-cost neural acquisition platform based on event-detection, addresses both the data and cost bottlenecks. In this architecture, action potentials are detected directly in analog hardware via per-channel threshold comparators and encoded as discrete digital timestamps, discarding the raw waveform entirely. The FPGA-based digital backend, implemented on a Lattice MachXO2-7000HE device, monitors all 60 channels simultaneously through a fully parallel processing pipeline, achieving a detection latency independent of the number of concurrently active channels, and reducing the average output data rate to approximately 12 kB/s, a reduction exceeding 99% relative to conventional continuous-sampling systems.

Threshold calibration is fully automated. Integrated ADCs acquire background noise statistics which are sent to the FPGA via the SPI protocol; a hybrid algorithm running on the host computer estimates the detection thresholds for each channel by applying a robust noise estimator primarily based on Quiroga’s MAD method, and the results are programmed into the integrated DACs via FPGA-controlled SPI state machines. A MATLAB application integrates the system, overseeing calibration, configuration, real-time decoding, and visualization.

Experimental validations confirm F1 scores exceeding 90% under nominal operating conditions, with robust performance maintained across more adverse scenarios including sub-calibration spike amplitudes, low signal-to-noise ratios, and high-rate Poisson spike trains. Onboard latency characterization further shows that the digital backend contributes

negligibly to the overall system response, with end-to-end detection latency remaining within a few hundred microseconds under high-SNR calibration, consistent with values reported for comparable closed-loop acquisition systems in the literature.

Although the event-driven modality is not intended for experiments where spike morphology is fundamental, it provides an accessible and scalable solution for applications where spatiotemporal patterns of neural activity constitute the key information, including studies on network synchronization, burst detection, pharmacological testing, and long-term closed-loop protocols.

Acknowledgements

Contents

List of Tables	8
List of Figures	9
I Introduction	11
1 Introduction	13
1.1 Aim of the project	13
1.2 Thesis Organization	14
II Background	17
2 Background and State of the Art	19
2.1 Neurophysiological background	19
2.1.1 Neuron electrophysiology	19
2.1.2 Action potentials and extracellular spikes	20
2.1.3 Synaptic transmission and network dynamics	23
2.2 In vitro neuronal cultures and MEA platforms	24
2.2.1 Cell cultures for electrophysiology	24
2.2.2 Multielectrode arrays (MEAs)	24
2.2.3 Noise sources, SNR, and system calibration	26
2.3 State of the Art in Neural Acquisition Systems	27
2.3.1 Continuous-sampling architectures	27
2.3.2 Commercial platforms and literature review	28
2.4 Moving from MCU to FPGA: Hardware Architecture	29
2.4.1 Limitations of sequential processing in microcontrollers	29
2.4.2 Why FPGA: general characteristics	29
III System Design and Implementation	31
3 System Architecture Overview	33
3.1 Platform Overview	33
3.1.1 Physical architecture	33

3.1.2	Signal path	35
3.1.3	Operational modes	35
3.2	Hardware Platform	36
3.2.1	MEA interface and analog front-end	37
3.2.2	Motherboard and event-detection chain	37
3.3	Event-Driven Acquisition Paradigm	38
3.3.1	Sampling window and sticky accumulation	39
3.3.2	Packet format	39
3.3.3	Data-rate analysis	39
3.4	Development Environment and Design Constraints	40
3.4.1	Target device and toolchain	40
3.4.2	FPGA interface summary	40
4	FPGA Implementation	43
4.1	Hardware Architecture and Module Hierarchy	43
4.1.1	Operational modes and signal multiplexing	44
4.2	Top-Level Module: <code>main_module</code>	45
4.2.1	Dual-source reset	45
4.2.2	Timestamp counter	45
4.2.3	Packet assembly and parity	47
4.2.4	Sticky mask reset	47
4.2.5	TX and CS multiplexers	47
4.3	Calibration Pipeline: <code>calibration_control</code>	48
4.3.1	Control protocol	48
4.3.2	Channel addressing scheme	49
4.3.3	ADC sampling phase	50
4.3.4	Threshold reception	51
4.3.5	DAC programming phase	52
4.3.6	SPI mode switching	52
4.4	SPI Master: <code>custom_spi_master</code>	52
4.4.1	Module parameters	52
4.4.2	FSM state description	53
4.4.3	Handshake interface	55
4.5	Event Detection: <code>edge_detector</code>	55
4.5.1	Synchroniser and glitch-filter chain	56
4.5.2	Majority-vote debouncing	57
4.5.3	Rising-edge detection	57
4.5.4	Sticky-mask accumulation	58
4.6	Datalogger: <code>WR_module</code>	58
4.6.1	Write process	59
4.6.2	Read process	59
4.7	UART Interface Stack	60
4.7.1	Transmitter: <code>uart_tx</code>	60
4.7.2	Receiver: <code>uart_rx</code>	61
4.7.3	Packet serialiser: <code>uart_fsm</code>	62
4.7.4	Command decoder: <code>uart_config_decoder</code>	64

5	Host Software: MATLAB GUI	67
5.1	Session Setup and Calibration	67
5.2	Threshold Algorithm	68
5.3	Asynchronous Event Acquisition	69
5.4	Refractory period filter	70
5.5	Real-Time Visualization	70
IV	Experimental Protocol and Validation Methods	71
6	Experimental Protocol	73
6.1	Hardware Setup and Signal Injection	73
6.2	Synthetic Spike Model and Acquisition Structure	74
6.2.1	Synthetic Spike Model	74
6.2.2	Alignment preamble and Acquisition Structure	75
6.3	Experimental Design	75
6.3.1	Study 1 — $K \times \text{SNR} \times f_r$ Sweep	76
6.3.2	Study 2 — Amplitude Sweep with Fixed Calibration	76
6.3.3	Study 3 — Poisson Firing Rate	77
6.3.4	Study 4 — Onboard Latency Characterization	78
6.4	Temporal Alignment and Clock Skew Correction	81
7	Validation Methods	85
7.1	Evaluation Metrics	85
7.2	Statistical Analysis	86
7.2.1	Study 1	87
7.2.2	Study 4	88
V	Results, Discussion and Conclusions	91
8	Results	93
8.1	Study 1 — $K \times \text{SNR} \times f_r$ Sweep	93
8.1.1	Repeatability	93
8.1.2	Detection Performance	95
8.1.3	Statistical Significance and Effect Size	100
8.1.4	Temporal Accuracy	101
8.2	Study 2 — Amplitude Sweep with Fixed Calibration	103
8.3	Study 3 — Poisson Firing Rate	104
8.4	Study 4 — Onboard Latency Characterization	105
8.4.1	FPGA-Internal Latency	105
8.4.2	System Latency — Descriptive Statistics	106
8.4.3	System Latency — Effect of SNR (Friedman and Kendall's W)	109
8.4.4	System Latency — Effect of K_{Quiroga} (Wilcoxon signed-rank)	110

9 Discussion	113
9.1 The K/SNR Ratio	113
9.2 Optimal K Selection and Operational Range	115
9.3 Calibration Robustness: Study 1 vs Study 2	117
9.4 Robustness to Poisson Timing: Study 1 vs Study 3	118
9.5 System Latency Characterization: Study 4	119
9.6 Limitations	121
10 Conclusions	123
Bibliography	125

List of Tables

2.1	Comparison of representative MEA acquisition platforms.	28
2.2	MCU-FPGA comparison.	30
3.1	FPGA interface pin summary.	41
4.1	FPGA sub-module summary.	44
4.2	ADS7038-Q1 SPI response frame.	51
6.1	Temporal structure of a single acquisition.	75
6.2	Parameter grid for Study 1.	76
6.3	Parameter grid for Study 2.	77
6.4	Parameter grid for Study 3.	78
6.5	Measurement parameters for the FPGA-internal latency (τ_{FPGA}).	79
6.6	Parameter grid for the end-to-end system latency (τ_{system}).	80
8.1	Per-session F_1 values for conditions exceeding the 10% CV threshold	95
8.2	Detection performance for $K_{\text{Quiroga}} = 3$	97
8.3	Detection performance for $K_{\text{Quiroga}} = 4$	98
8.4	Detection performance for $K_{\text{Quiroga}} = 5$	99
8.5	Conditions with a statistically significant effect of K_{Quiroga} on F_1	101
8.6	Mean jitter per K_{Quiroga} and SNR	102
8.7	Detection performance and jitter as a function of spike amplitude A_{spike} . .	104
8.8	Detection performance and jitter as a function of mean firing rate	104
8.9	Per-session results for the FPGA-internal latency (τ_{FPGA}).	106
8.10	Per-session descriptive statistics of the system latency (τ_{system}) for $K_{\text{Quiroga}} =$ 3	107
8.11	Per-session descriptive statistics of the system latency (τ_{system}) for $K_{\text{Quiroga}} =$ 4	108
8.12	Friedman test results for the effect of SNR on median system latency	109
8.13	Wilcoxon signed-rank test results for the effect of K_{Quiroga} on median system latency	111

List of Figures

2.1	Equivalent circuit of the plasma membrane	20
2.2	Action Potential.	21
2.3	Extracellular spike waveforms and comparison with intracellular recordings.	22
2.4	Two main synapse types.	23
2.5	Primary neuronal cell culture	24
2.6	MEA example.	25
2.7	Randles equivalent circuit model for the electrode-electrolyte interface	26
2.8	FPGA architecture overview	30
3.1	Full system block diagram.	34
3.2	Single-channel signal path.	36
3.3	AFE PCB and complete platform.	37
4.1	Hardware architecture module hierarchy.	43
4.2	Operational flowchart of the NeuERA acquisition platform.	46
4.3	Sticky reset timing.	47
4.4	Complete calibration FSM (Lattice Diamond export).	48
4.5	High-level calibration FSM phases.	49
4.6	SPI master FSM — Lattice Diamond export.	53
4.7	SPI Mode 0 vs Mode 3 timing diagram.	55
4.8	Per-channel event detection datapath.	56
4.9	Majority-vote debouncing — example waveform.	58
4.10	uart_tx FSM (Lattice Diamond export).	61
4.11	uart_rx FSM (Lattice Diamond export).	62
4.12	uart_fsm FSM (Lattice Diamond export).	63
6.1	Experimental validation setup.	74
6.2	Example of the synthetic composite signal injected by the DAQ.	75
6.3	Example of the synthetic signal with Poisson-distributed inter-spike intervals.	78
6.4	Oscilloscope measurement of the FPGA-internal latency	79
6.5	Example of the DAQ-to-GPIO peak matching procedure	81
6.6	Temporal alignment result for a representative acquisition.	83
8.1	Coefficient of variation (CV) of the session-level F_1 -score across the SNR- f_r experimental space.	94
8.2	Grand-average F_1 -score as a function of the firing rate (f_r)	96
8.3	Grand-average F_1 -score as a function of the SNR	96
8.4	F_1 -score across the SNR- f_r space for each K_{Quiroga} value	100
8.5	Mean jitter ($\pm$ std) as a function of SNR	102

8.6	Pooled jitter distribution for each K_{Quiroga} value	102
8.7	F_1 -score as a function of spike amplitude A_{spike}	103
8.8	F_1 -score as a function of mean firing rate under Poisson spike timing . . .	105
8.9	Median system latency by SNR and K_{Quiroga} , with post-hoc pairwise significance	110
8.10	Median system latency by K_{Quiroga} and SNR, with pairwise significance . .	111
9.1	F_1 -score as a function of the K_{Quiroga} /SNR ratio	114
9.2	Representative calibration traces at $K_{\text{Quiroga}} = 4$, SNR = 6 ($K/\text{SNR} = 0.67$, transition zone)	115
9.3	Representative calibration traces at $f_r = 50$ Hz, $K_{\text{Quiroga}} = 4$ ($K/\text{SNR} = 1.33$, structural impossibility zone)	116
9.4	Representative calibration traces at $K_{\text{Quiroga}} = 4$, SNR = 10, $f_r = 300$ Hz .	117
9.5	Representative calibration traces at $K_{\text{Quiroga}} = 4$, SNR = 10, mean $f_r = 300$ Hz (Poisson timing)	119

Part I

Introduction

Chapter 1

Introduction

The study of in vitro neural cells has become a pillar in modern neuroscience, guaranteeing crucial insights into neurophysiology, network dynamics, and pharmacological testing moving beyond passive observation towards active, bidirectional interfaces. Studies have demonstrated that in vitro cortical neurons can exhibit embodied intelligence and goal-directed behavior—such as learning to play the video game Pong within a simulated environment[1]. Even though these advancements are promising, scaling such bio-hybrid systems leads to severe methodological and financial bottlenecks. Conventional micro-electrode array (MEA) acquisition systems are exceptionally expensive, with commercial 60-channel setups starting around 30,000 USD and advanced configurations easily exceeding hundreds of thousands of dollars. Moreover, maintaining the repeatability of these experiments while ensuring the survival of the cell cultures requires not only complex instrumentation but also highly specialized expertise[2]. A primary driver of these high costs and system complexities is the data management paradigm, where standard electrophysiological equipment usually relies on continuous, high-speed analog-to-digital conversion, typically sampling each channel at 10 kHz or higher to preserve the full waveform morphology of the neural spikes. While waveform fidelity is critical for basic electrophysiological research and single-unit spike sorting, many advanced applications, such as studying network synchronization[3], burst detection and closed-loop stimulation[4], and real-time behavioral closed-loop tasks[1, 5], primarily require only the spatiotemporal activation map.

1.1 Aim of the project

This thesis is part of a macro-project aimed at developing a low-cost, highly scalable platform for real-time neural signal acquisition. In order to bypass the massive data generation of traditional continuous sampling setups, the whole project adopts a hardware-level event-driven architecture, where action potentials are detected via analog thresholding directly on the motherboard and encoded as binary events (a digital "1" when a spike event occurs and "0" otherwise), rather than digitizing the raw signal. The previous digital backend of this platform relied on an ARM Cortex-M7 microcontroller (MCU), whose sequential processing nature had become a severe bottleneck as channel concurrency increased. The MCU-based architecture could safely manage a maximum of 15 simultaneous channels. The

main goal of this thesis work is to overcome the limitations of the MCU by designing and implementing a new digital acquisition architecture based on a Lattice Field-Programmable Gate Array (FPGA), coupled with a custom host-side software interface. By leveraging the inherent parallel computing capabilities of FPGAs, the proposed system aims to manage all 60 channels simultaneously while overseeing the entire hardware setup. Specifically, the project focuses on three main parts:

1. **Automated Hardware Calibration Pipeline:** Designing a finite state machine (FSM) to manage the Analog-to-digital Converters (ADCs) sampling pipeline to acquire baseline background noise, streaming the raw data to the host PC for threshold calculation, and receiving and programming the computed reference voltages into the Digital-to-Analog Converters (DACs) to configure the analog triggers.
2. **Parallel Event Acquisition and Zero-Loss Streaming:** Implementing a fully parallel digital backend that captures 60 independent event lines simultaneously and sends it to the host PC.
3. **Host-Side Real-Time Visualization and Control:** Developing a custom Graphical User Interface (GUI) to manage the bidirectional communication, acting as an asynchronous Logic Analyzer to provide a real-time visualization of the neural network's spatiotemporal activity.

1.2 Thesis Organization

Beyond the present introductory chapter, the thesis is structured into four additional parts:

- **Part II** provides the necessary theoretical background (Chapter 2), focusing on the neurophysiological basis of extracellular signal acquisition, the role of microelectrode arrays (MEAs) in in vitro studies, and the state of the art in neural acquisition systems, highlighting the motivation for developing detection systems such as the one proposed in the present work.
- **Part III** describes the design and implementation of the system across three chapters. Chapter 3 provides an overview of the device, focusing primarily on its hardware implementation; Chapter 4 describes in detail the FPGA RTL design, focusing on the implementation logic that led to the development of the hardware architecture; Chapter 5 describes the host-side MATLAB graphical user interface developed that manages the entire data acquisition process.
- **Part IV** is dedicated to the experimental and validation setup across two chapters. Chapter 6 describes the experimental setup and the studies designed to evaluate different characteristics of the overall system; Chapter 7 defines the evaluation metrics and the statistical analysis methods employed.
- **Part V** is structured into three chapters presenting the results obtained. Chapter 8 presents the experimental results from the four validation studies, objectively evaluating the obtained parameters from a mathematical standpoint; Chapter 9 discusses

their implications in terms of detection performance, system limitations, and robustness, formulating questions arising from the results and providing answers based on the knowledge acquired during the system characterisation; finally, Chapter [10](#) presents the final conclusions of the work and outlines directions for future development.

Part II

Background

Chapter 2

Background and State of the Art

2.1 Neurophysiological background

2.1.1 Neuron electrophysiology

Neurons are the fundamental units of the nervous system. From an electrical point of view, the neuronal plasma membrane acts as a dielectric insulator, separating two conductive electrolytic media: the intracellular (ICL) and extracellular (ECL) fluids, providing the membrane with an intrinsic electrical capacitance (C_m), allowing it to store charge.

The neuronal membrane contains specialized protein structures called ion channels, which selectively allow specific ions (Na^+ , K^+ , Ca^{2+} , and Cl^-) to cross it. Since these ions are not uniformly distributed between the intracellular and extracellular compartments, a voltage difference arises across the membrane, defined as the membrane potential (V_m). When V_m becomes less negative than its resting value the membrane is depolarized; when it becomes more negative, it is hyperpolarized.

The steady-state resting membrane potential of a typical neuron is approximately -70 mV and it is accurately described by the Goldman-Hodgkin-Katz (GHK) voltage equation model [6, 7], also described with an equivalent circuit illustrated in Figure 2.1, that demonstrates how the membrane potential depends on the relative permeabilities (P) and the intra- (i) and extracellular (o) concentrations of the primary ions:

$$V_m = \frac{RT}{F} \ln \left(\frac{P_K[K^+]_o + P_{Na}[Na^+]_o + P_{Cl}[Cl^-]_i}{P_K[K^+]_i + P_{Na}[Na^+]_i + P_{Cl}[Cl^-]_o} \right) \quad (2.1)$$

where R is the ideal gas constant, T is the absolute temperature, and F is the Faraday constant. In mammalian neuronal cells, the resting membrane permeability to chloride is often severely limited compared to potassium, allowing the equation to be practically approximated using only Na^+ and K^+ dynamics[8]. To maintain the thermodynamic disequilibrium, the cell continuously expends metabolic energy (ATP) via the Na^+/K^+ pump, extruding, for example, three Na^+ ions while importing two K^+ ions against their respective gradients, restoring the ionic balance and acting as a constant, active current source.

Integrating these components into the original model, the neuronal membrane can be mathematically formalized as a parallel electrical circuit consisting of a capacitor and variable conductances. The total transmembrane current (I_{tot}) is the sum of the capacitive displacement current and the resistive ionic currents:

$$I_{tot} = C_m \frac{dV_m}{dt} + \sum_s g_s (V_m - E_s) \quad (2.2)$$

where g_s represents the macroscopic conductance of a specific ion channel s , and E_s is its Nernst equilibrium potential. The term $(V_m - E_s)$ constitutes the electrochemical driving force acting on that ion. In a true resting state, the sum of all ionic fluxes is zero ($I_{tot} = 0$), maintaining the neuron in a metastable, polarized condition ready to discharge when adequately stimulated.

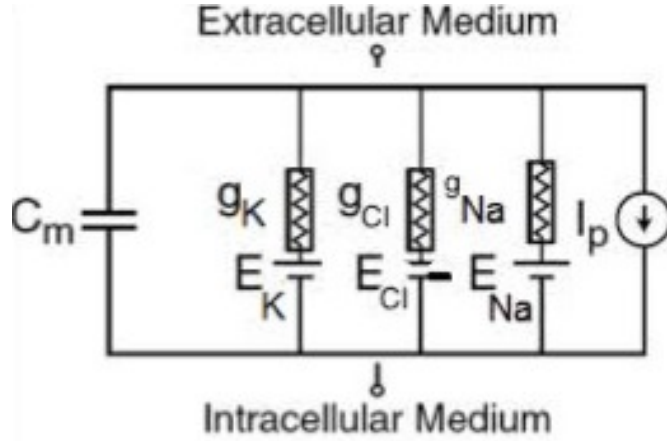


Figure 2.1. Equivalent circuit of the plasma membrane.

2.1.2 Action potentials and extracellular spikes

While the resting membrane potential represents a stable equilibrium, excitable cells like neurons can actively modulate their membrane permeability in response to external stimuli. In fact, if a depolarizing stimulus raises the membrane potential above a specific threshold (typically around -45 mV), it triggers a rapid, nonlinear physiological response known as the action potential (Figure 2.2).

The action potential was mathematically described by Hodgkin and Huxley [9]. By extending the parallel RC circuit model introduced in the previous section, it has been demonstrated that the macroscopic membrane conductances for sodium (g_{Na}) and potassium (g_K) are not strictly constant, but rather highly non-linear functions of both voltage and time. Therefore, the total transmembrane current during an action potential can be described by the Hodgkin-Huxley equation:

$$I_{tot} = C_m \frac{dV_m}{dt} + \bar{g}_K n^4 (V_m - E_K) + \bar{g}_{Na} m^3 h (V_m - E_{Na}) + \bar{g}_l (V_m - E_l) \quad (2.3)$$

where $\bar{g}$ represents the maximal conductance for each ion type, while $\bar{g}_l$ and E_l represent the passive leakage components and the dimensionless gating variables m , h , and n (ranging from 0 to 1) describe the probability of the channels being open.

The generation of an action potential follows a strict all-or-none principle and unfolds in three phases:

1. **Depolarization:** Crossing the voltage threshold causes the activation gates (m) of the Na^+ channels to open, triggering a positive feedback loop that drives the membrane potential rapidly toward the sodium equilibrium potential ($E_{\text{Na}} \approx +58$ mV). The intracellular potential increases from -70 mV to a peak of approximately +40 mV.
2. **Repolarization:** Depolarization terminates due to the intrinsic closure of the Na^+ inactivation gates (h), which blocks further sodium influx, and the delayed opening of voltage-gated K^+ channels (n). The resulting efflux of potassium rapidly repolarizes the cell, driving V_m back down.
3. **Hyperpolarization and Refractory Period:** K^+ channels close slowly, so the membrane briefly undershoots the resting potential, resulting in a refractory period lasting approximately 1 ms, during which Na^+ channels remain inactivated, making it impossible to generate another action potential. This physiological constraint strictly limits the maximum firing frequency of a neuron to about 1000 Hz and ensures the unidirectional propagation of the signal along the axon.

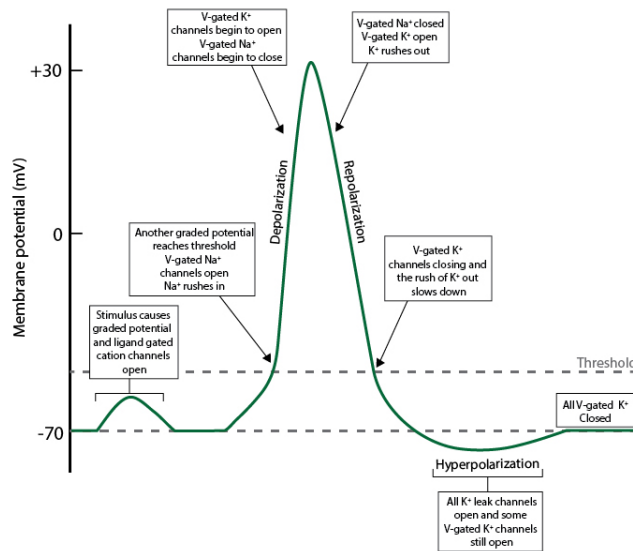


Figure 2.2. **Action Potential.** Plot of an action potential generated by simulation of the Hodgkin–Huxley model outlining different phases.

The resulting signal does not remain confined within the cell membrane, but rather propagates through the surrounding extracellular medium, generating a potential field of

lower amplitude than the intracellular action potential, yet still measurable by advanced systems such as the electrodes of microelectrode arrays (MEAs) (Section 2.2.2).

The attenuation of signal amplitude is mainly attributable to the properties of neuronal tissue, which, from an electrical point of view, consists of cells embedded in a low-resistance extracellular medium, occupying less than one fifth of the total volume. As a result, adjacent cells are largely electrically decoupled and the extracellular potential differences recordable at distinct locations remain small, typically below one millivolt [10], in contrast with the transmembrane potential, which reaches values between 50 and 100 mV during an action potential.

The extracellular spike waveform is typically biphasic, usually describable as an initial negative deflection followed by a smaller positive rebound (Figure 2.3), with an amplitude typically ranging from 10 to 500 μV that tends to grow with the size of the soma and apical dendrites, as is notably the case for large layer-5 pyramidal neurons [11, 12], and a duration typically below 2 ms [13, 14]. Furthermore, the recorded amplitude is strongly modulated by the position of the electrode relative to the neuron: signals are largest near the perisomatic region and decay rapidly with distance, falling below 20 μV at separations exceeding $\sim 100 \mu\text{m}$ [14, 15]. Moreover, in vitro neuronal networks exhibit spontaneous firing activity with frequencies typically distributed between 0.05 and 30 Hz [3].

A crucial aspect is that extracellular recording methods provide information mostly on whether a neuron generates a spike or not. Consequently, the signal detection process can be inherently treated as a discrete binary classification task, which perfectly aligns with the operational paradigm of our proposed device.

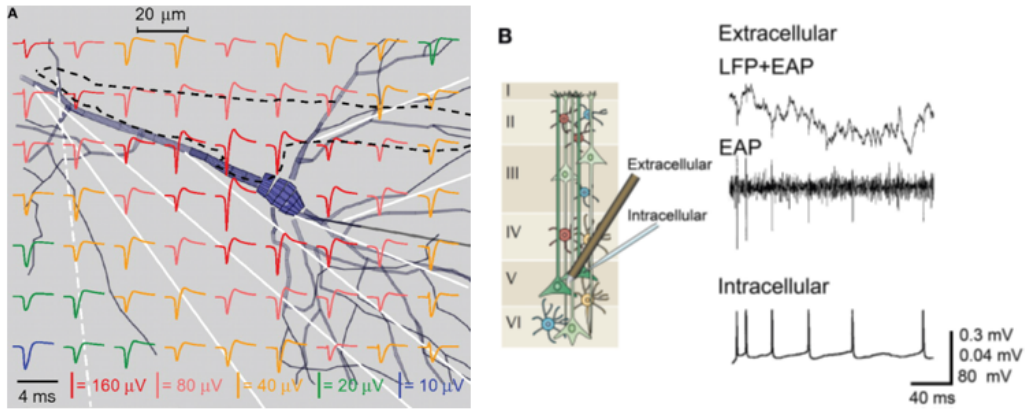


Figure 2.3. Extracellular spike waveforms. (A) Simulated EAP spatial map around a CA1 (Cornu Ammonis area 1, hippocampus) pyramidal neuron, showing the color-coded amplitude decay with distance from the soma (red: 160 μV ; blue: 10 μV). Scale bars: 20 μm , 4 ms. Adapted from [15] via [14]. (B) Simultaneous extracellular (EAP, 0.04 mV scale) and intracellular (IAP, 80 mV scale) recordings from a cortical pyramidal cell, illustrating the amplitude contrast between the two recording modalities. Adapted from [14].

2.1.3 Synaptic transmission and network dynamics

The complexity of the nervous system comes from the interconnectedness of neurons. In fact, information transfer between cells occurs at specialized junctions known as synapses, which are primarily classified as electrical, chemical, or mixed (Figure 2.4). In mammalian nervous systems, chemical synapses are the predominant mode of inter-neuronal communication [16].

When an action potential propagates down the axon and reaches the presynaptic terminal, the localized membrane depolarization triggers the opening of voltage-gated calcium (Ca^{2+}) channels: ions rush into the intracellular space and activate calcium-sensor proteins, which interact with the protein complex to mediate the fusion of synaptic vesicles with the presynaptic membrane [16]. The released neurotransmitter molecules diffuse across the synaptic cleft and bind to specific receptors on the postsynaptic membrane.

Postsynaptic receptors are broadly classified into two primary macrofamilies: ionotropic and metabotropic [17]. Ionotropic receptors are ligand-gated ion channels that directly alter membrane permeability upon neurotransmitter binding, mediating fast synaptic transmission. Instead, metabotropic receptors initiate slower, neuromodulatory responses via intracellular G-protein-coupled second-messenger cascades [17].

Depending on which ion channels are recruited by receptor activation, the postsynaptic membrane potential can shift in either direction. In fact, cation influx (Na , Ca^{2+}) depolarizes the membrane, producing an Excitatory Postsynaptic Potential (EPSP) that pushes V_m closer to the firing threshold, while anion influx (Cl) or K efflux has the opposite effect, generating an Inhibitory Postsynaptic Potential (IPSP) that reduces the probability of firing [18]. Whether a new action potential is ultimately triggered depends on the spatial and temporal summation of all incoming EPSPs and IPSPs: if the net depolarization at the axon hillock exceeds threshold, the neuron fires; otherwise, it remains silent. When plated on a Microelectrode Array (MEA), neurons in vitro self-organize into functional networks over the course of a few weeks. At the network level, the dominant activity pattern is bursting: short windows of intense, synchronized firing that recur spontaneously and involve a large fraction of the population.

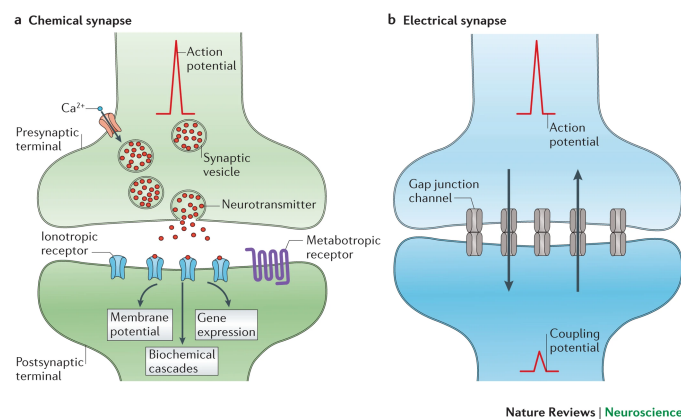


Figure 2.4. **Two main synapse types.** (a) Chemical synapse mediated by neurotransmitters. (b) Electrical synapse via gap junctions.[19]

2.2 In vitro neuronal cultures and MEA platforms

2.2.1 Cell cultures for electrophysiology

Cell culture refers to the isolation and maintenance of cells *in vitro* under strictly controlled environmental conditions, serving as a model system for biological, medical, and pharmacological research [20]. These cultures require a tightly regulated artificial environment, involving the use of specific nutrient media supplemented with growth factors, precise temperature regulation and a controlled gas mixture to maintain a stable physiological pH [21, 22].

In neuroscience, *in vitro* neuronal cultures represent a compromise between the extreme systemic complexity of *in vivo* animal models and the oversimplification of isolated single-cell studies. When plated on an appropriate substrate, cultured neurons are capable of spontaneously self-organizing, extending neurites, and forming highly interconnected, functional synaptic networks, ideally replicating the electrophysiological dynamics and bursting behaviors observed in intact neural tissues [23].

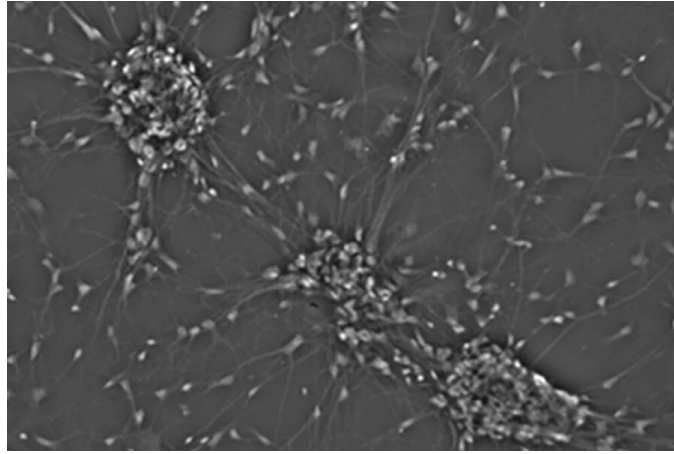


Figure 2.5. **Primary neuronal cell culture example**[24].

Traditional intracellular recording techniques, such as the patch-clamp method [25], provide exceptional signal-to-noise ratios but are inherently invasive and inevitably rupture the cell membrane, limiting the experiments to acute, short-term observations (typically a few hours) of single neurons. These limitations have made it necessary to fully exploit the potential of *in vitro* networks and observe their long-term spatiotemporal evolution using multi-site recording technologies, such as Microelectrode Arrays (MEAs).

2.2.2 Multielectrode arrays (MEAs)

The scientific community has widely adopted planar Microelectrode Arrays (MEAs), which are hybrid bioelectronic devices that integrate a cell culture chamber directly on top of an array of embedded micro-sensors [26].

The base substrate of a standard planar MEA is typically composed of a glass wafer; microelectrodes are photolithographically patterned onto this substrate and fabricated using noble metals or highly stable compounds, such as titanium nitride (TiN), platinum (Pt), gold (Au), or iridium oxide (IrO_x) [14]. The entire surface, except for the active electrode sites and the peripheral contact pads, is then passivated with an insulating layer to confine the sensing area and prevent electrical shorts [26]. One of the most important characteristics of MEAs is achieving the lowest possible electrode impedance, which unfortunately creates a critical design trade-off: a higher spatial density improves the spatial resolution of the network mapping but requires a smaller electrode surface area. On the other hand, reducing the area increases the electrical impedance [12], rendering the sensor highly susceptible to thermal noise. In fact, modern electrode design focuses on maximizing the effective surface area without increasing the geometric footprint. The employment of porous conductive structures such as carbon nanotubes, graphene, and doped diamond allows for a drastic decrease in electrode impedance, significantly improving the Signal to Noise Ratio (SNR) while maintaining high spatial resolution [14].

MEA platform architecture is rapidly evolving towards Field-Effect Transistor (FET) and Complementary Metal-Oxide-Semiconductor (CMOS) technologies which are increasingly exploited to fabricate High-Density MEAs (HDMEAs), capable of mapping a neuronal network at sub-cellular resolution [14].

Another innovation is integrating the analog readout circuitry directly onto the sensing substrate, minimizing parasitic capacitances and resistances, significantly improving signal recording quality, and allowing for active, on-chip signal multiplexing.

Even though research is rapidly growing, the standard 60-channel passive MEA remains the gold standard for many macroscopic network-level and toxicological studies due to its robustness, accessibility, and excellent long-term stability [27, 14]. However, unlike CMOS arrays that handle multiplexing internally, passive MEAs stream all 60 analog channels continuously and simultaneously.

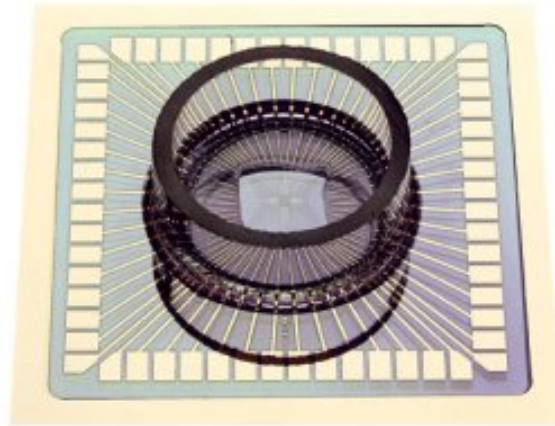


Figure 2.6. **MEA example.** Example of Multi-Electrode Arrays [28]

2.2.3 Noise sources, SNR, and system calibration

The successful extraction of microscopic extracellular action potentials in the majority relies on the ability of the acquisition system to distinguish these transient events from the background noise floor, which in a well-designed passive MEA system typically lies between 5 and 20 μV_{rms} [14]. When a metallic microelectrode is immersed in the saline culture medium, a complex electrochemical boundary layer forms at the metal-liquid junction and, according to the Randles equivalent circuit model (Figure 2.7), this interface can be represented by a double-layer capacitance (C_{dl}) in parallel with a charge-transfer resistance (R_{ct}), both in series with the spreading resistance of the bulk solution (R_s) [29]. Because these components possess a real resistive part, they intrinsically generate Johnson-Nyquist (thermal) noise, whose power spectral density is directly proportional to the real part of the electrode impedance $\text{Re}[Z_{el}(f)]$ and the absolute temperature T [30].

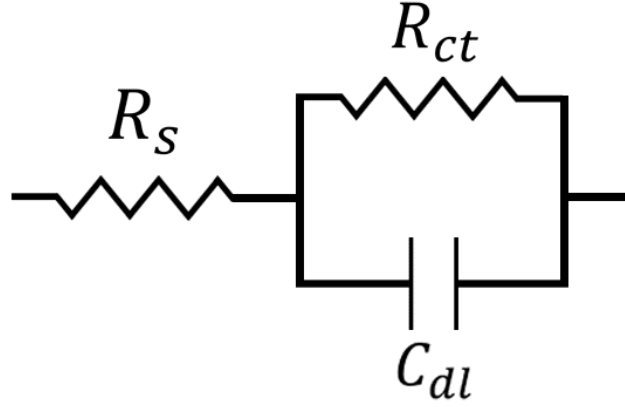


Figure 2.7. **Randles equivalent circuit model.** The electrical equivalent circuit of the electrode-electrolyte interface, comprising the spreading resistance of the bulk solution (R_s) in series with the parallel combination of the charge-transfer resistance (R_{ct}) and the double-layer capacitance (C_{dl}).

Before being digitized, the raw analog signals must be amplified and band-limited. The Low-Noise Amplifier (LNA) at the front of the acquisition chain introduces its own intrinsic electronic noise, which must be kept strictly below the electrode noise floor to avoid degrading the overall SNR, requiring a fundamental trade-off between minimizing the input-referred noise — typically below 10 μV_{rms} [31] — and constraining power consumption to prevent thermal damage to the culture. A further contribution comes from $1/f$ (flicker) noise intrinsic to the active devices in the amplifier front-end, which dominates below a few hundred hertz [31]. In order to suppress this low-frequency component, the signal is usually band-pass filtered with a high-pass corner around 300 Hz and a low-pass corner around 3–5 kHz, which isolates the frequency band of extracellular action potentials, removes out-of-band components that would otherwise alias during digitization [14], and ensures the signal is zero-mean. The overall recording quality is quantified by the SNR, defined in extracellular electrophysiology as the ratio between the peak-to-peak amplitude of the spike (V_{pp}) and the RMS value of the background noise (σ_n) [14].

A complex challenge arises from the intrinsic heterogeneity of *in vitro* cultures, since

neuronal adhesion, glial cell proliferation, and electrode degradation vary randomly across the array. It occurs because the baseline noise σ_n is uniquely different for each of the 60 channels. Applying a single global threshold across the entire array is therefore highly inefficient: a value set too low triggers continuous false positives on noisy channels, while one set too high misses physiological spikes on quieter ones. These considerations together reinforce the necessity of performing calibration *in situ*, with the electrodes already immersed in physiological solution and the full acquisition chain active, rather than relying on tabulated noise values and instead dynamically estimating a channel-specific noise floor. The standard solution in neural signal processing, whether the downstream processing operates on full spike waveforms or binary event timestamps, is the Median Absolute Deviation (MAD) estimator proposed by Quiroga et al. [32], which provides a spike-insensitive estimate of σ_n . For a zero-mean signal $x(t)$, it is defined as:

$$\sigma_n = \frac{\text{median}(|x|)}{0.6745} \quad (2.4)$$

where the constant 0.6745 corresponds to the MAD of a unit-variance Gaussian distribution, making σ_n a consistent and asymptotically efficient estimate of the true noise standard deviation regardless of the presence of sparse spike events. The per-channel detection threshold is then set as a fixed multiple of σ_n , typically $k \cdot \sigma_n$ with $k \in [4, 5]$, chosen as a compromise between detection sensitivity and false positive rate [32].

2.3 State of the Art in Neural Acquisition Systems

2.3.1 Continuous-sampling architectures

The commercial landscape for *in vitro* MEA acquisition systems can be stratified into different tiers, each reflecting a different trade-off between spatial resolution, throughput, and cost, sharing a common architectural paradigm: continuous, wideband digitization of every channel at sampling rates typically between 10 and 50 kHz.

At the entry level of professional laboratory instrumentation, the MCS MEA2100 (Multi Channel Systems, Reutlingen, Germany) represents the historical gold standard for passive 60-channel MEA recordings. The system supports simultaneous acquisition from 60, 120, or 256 electrodes at up to 50 kHz per channel, with integrated analog stimulation and temperature control [33]. A one-hour continuous recording session at 20 kHz and 16-bit resolution from 60 channels alone generates GBs of raw data, a figure that scales linearly with session duration and channel count, placing significant demands on storage infrastructure and offline analysis pipelines. Another solution, developed by Maxwell Biosystems (Zürich, Switzerland), is a CMOS-based High-Density MEA (HD-MEA) platform derived from the academic prototype [34]. The MaxOne system integrates 26,400 electrodes at a pitch of $17.5 \mu\text{m}$ within a $3.85 \times 2.10 \text{ mm}^2$ sensing area, enabling simultaneous sub-cellular and network-level recordings. To manage the resulting data volume, the system employs a reconfigurable switch-matrix architecture that routes any arbitrary subset of electrodes to a fixed pool of 1,024 readout channels at 20 kHz [34].

For high-throughput pharmacological screening, Axion BioSystems (Atlanta, USA) offers the Maestro Pro platform, capable of recording from up to 96 wells simultaneously through 768 total channels [35]. The system still operates on a continuous-sampling

paradigm, requiring dedicated server-grade computing infrastructure to process and store the resulting data streams in real time.

The analog front-end electronics underpinning the majority of these systems — including many academic custom solutions — rely on the Intan RHD series of integrated amplifier-ADC chips [36]. These chips perform per-channel amplification, bandpass filtering, and 16-bit digitization at up to 30 kHz, streaming the resulting data over SPI to a host FPGA. Intan’s own reference implementation — the open-source *Rhythm* interface — uses a Xilinx FPGA to aggregate up to 256 channels over parallel SPI buses and forward the data to a host PC via USB [37].

2.3.2 Commercial platforms and literature review

Table 2.1 summarises the key specifications of the representative commercial platforms discussed above, alongside the low-cost academic system presented by Garma et al. [38] based on Intan RHD2132 chips.

System	Channels	Fs (kHz)	Res. (bit)	Paradigm	Cost (est.)
MCS MEA2100 [33]	60–256	50	16	Continuous	\$30–50k
Maxwell MaxOne [34]	1,024*	20	10	Continuous	>\$100k
Axion Maestro Pro [35]	768	12.5	16	Continuous	\$50–80k
Intan RHD + custom PCB [38]	60	30	16	Continuous	~\$1k

Table 2.1. Comparison of representative MEA acquisition platforms. *Switch-matrix: only 1,024 of 26,400 electrodes routed simultaneously. ** Costs are indicative estimates based on publicly available data and vendor communications; exact pricing is available upon request from respective manufacturers.

The data in Table 2.1 make explicit the cost and complexity gradient that characterises the existing landscape. All commercial platforms adopt continuous waveform acquisition, which provides maximum analytical flexibility at the price of generating data volumes that scale with session duration and channel count. Even the lowest-cost academic alternative [38], while reducing the hardware expenditure by nearly two orders of magnitude with respect to commercial solutions, inherits the same continuous-sampling paradigm and its associated data management burden.

For applications that do not require full waveform reconstruction, such as network-level burst detection, synchrony quantification, and real-time closed-loop stimulation, this overhead is entirely avoidable. The elimination of redundant background samples at the source, retaining only the binary occurrence and hardware timestamp of threshold-crossing events, is the core motivating principle of the event-driven architecture. This paradigm shift compresses the data rate from GBs per hour generated by a standard 60-channel continuous system down to a few kilobytes per second during typical *in vitro* network activity.

2.4 Moving from MCU to FPGA: Hardware Architecture

2.4.1 Limitations of sequential processing in microcontrollers

While MCUs are ideal for general-purpose embedded control tasks, their fundamental constraint lies in their sequential execution model, because their processor core handles one instruction at a time, and concurrency must be emulated through interrupt service routines (ISRs) or polling loops. In the context of a 60-channel MEA, this constraint becomes the primary architectural bottleneck during synchronized network bursts. When multiple electrodes detect an action potential simultaneously, the analog comparators generate concurrent hardware interrupts. The MCU must line up these requests and service them one at a time. The first channel to be serviced receives a precise timestamp; every subsequent channel accumulates an additional delay equal to the ISR execution time. On a Cortex-M7 at 216 MHz, interrupt entry overhead alone requires approximately 12 clock cycles (≈ 55 ns) [39], and a minimal ISR including timestamp capture and data storage is conservatively estimated at 1–2 μ s for a C implementation. Servicing all 60 channels sequentially therefore introduces a worst-case jitter of:

$$\Delta t_{\max} = (N - 1) \times t_{\text{ISR}} \approx 59 \times 2 \mu\text{s} = 118 \mu\text{s} \quad (2.5)$$

where $N = 60$ is the channel count and t_{ISR} is the per-channel service time. For neural signals with millisecond-scale temporal precision requirements, this accumulated jitter is unacceptable. Moreover, during dense burst activity where firing rates can transiently reach hundreds of events per second across the entire array, the interrupt queue saturates and events are irreversibly lost. These constraints limited the previous MCU-based implementation to a maximum of 15 simultaneous channels before timing integrity degraded below acceptable levels.

2.4.2 Why FPGA: general characteristics

To overcome these limitations, this thesis adopts a Field-Programmable Gate Array (FPGA) as the digital backend. Unlike an MCU, an FPGA does not execute a sequential instruction stream. Its programmable fabric, consisting of configurable logic blocks (CLBs), lookup tables (LUTs), flip-flops, and dedicated routing resources (Figure 2.8) [40], allows arbitrary digital circuits to be instantiated directly in hardware. Since each circuit element operates concurrently with all others, driven by a common clock, applying this architecture to the 60-channel acquisition problem means that 60 independent timestamping modules can be instantiated in parallel on the FPGA fabric. Every channel is monitored within the same clock cycle, and a threshold-crossing event on any subset of channels is captured simultaneously with no queuing overhead. The execution time is therefore $O(1)$ and independent of the number of channels firing concurrently, in contrast to the $O(N)$ scaling of the interrupt-driven MCU approach. As demonstrated by Hassan et al. [41], this parallelism is precisely what makes FPGA-based platforms effective for the concurrent processing of high-density MEA signals in real time.

Table 2.2 summarises the key architectural differences between the MCU-based predecessor and the FPGA-based system proposed in this thesis.

Property	MCU (ARM Cortex-M7)	FPGA (Lattice MachXO2)
Execution model	Sequential (ISR-based)	Parallel (spatial hardware)
Channel scalability	$O(N)$	$O(1)$
Max simultaneous channels	15	60 and more
Timestamping jitter	$\sim 100 \mu\text{s}$	< 1 clock cycle ($\sim 20 \text{ ns}$)
Worst-case latency	Variable (ISR-dependent)	Deterministic
Closed-loop capable	Yes (limited by ISR latency)	Yes (deterministic latency)
Reconfigurability	Firmware (recompile)	Bitstream (reprogram)

Table 2.2. Architectural comparison between the MCU-based predecessor system and the FPGA-based architecture proposed in this thesis.

Motivated by the requirements outlined above, this thesis proposes a fully redesigned digital backend centred on a Lattice MachXO2-7000HE FPGA. The system is structured around functional blocks, which are described in detail in the following chapters.

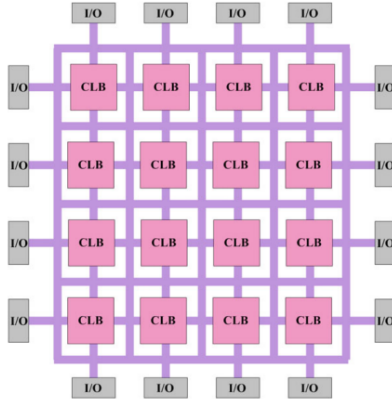


Figure 2.8. **FPGA architecture overview.** Configurable Logic Blocks (CLBs) are arranged in a two-dimensional grid and interconnected by programmable routing resources, with Input/Output (I/O) blocks positioned at the periphery [40].

Part III

**System Design and
Implementation**

Chapter 3

System Architecture Overview

3.1 Platform Overview

The platform described in this thesis is designed for real-time, event-driven acquisition of neural spiking activity from up to 60 electrode channels, interfacing with standard passive 60-channel microelectrode arrays (MEAs). The detection is based on an event-driven paradigm, which consists in detecting threshold-crossing events directly in analog hardware and encoding them as discrete digital pulses, rather than digitizing the raw extracellular waveform on each channel. This event-driven paradigm, introduced by the predecessor MCU-based platform [42], eliminates the need for high-speed analog-to-digital conversion across 60 channels and reduces the output data rate by more than 99% compared to conventional continuous-sampling systems operating at 10 kS/s per channel.

The present work redesigns the digital backend of the platform around a Lattice MachXO2-7000HE Field-Programmable Gate Array (FPGA), replacing the ARM Cortex-M7 microcontroller that previously limited the system to 15 simultaneous channels. The analog signal conditioning chain and the motherboard hardware are inherited from the prior implementation [42] and are described here to the extent required to characterize the operating boundaries of the digital system; their circuit-level design falls outside the scope of this work, while the FPGA implementation and the companion MATLAB host interface constitute the primary contributions of this thesis.

3.1.1 Physical architecture

The platform is organized into five functionally distinct blocks, arranged in a linear signal-processing chain from the biological preparation to the host computer, illustrated in Figure 3.1.

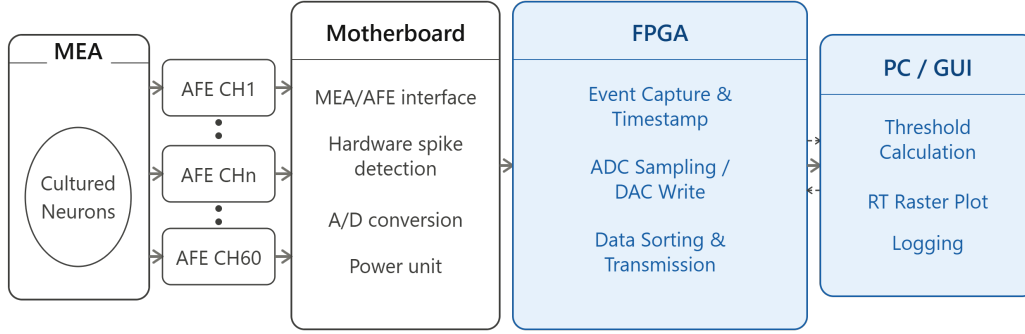


Figure 3.1. **Full system block diagram.** The five-block signal chain from the MEA to the host PC. The FPGA digital backend and the MATLAB GUI, highlighted in blue, constitute the contributions of this thesis.

MEA interface. In order to provide electrical access to the 60 recording sites and the common reference electrode, the platform interfaces with standard passive 60-channel MEAs through spring-loaded contacts (Section 3.2).

Analog Front-End (AFE). Since the voltage generated by individual neuronal cells is on the order of microvolts, each of the 60 signal lines is conditioned by a dedicated AFE board, which performs both low-noise amplification and band-pass filtering before reaching the motherboard for event detection (Section 3.2). As described in Section 6.1, this unit was replaced with a passive bridge during the experimental validation, for the reasons discussed there.

Motherboard. The motherboard is the central hardware unit of the platform: it provides regulated power to all modules, scales each conditioned signal via a PGA, and implements per-channel threshold comparison through a Schmitt trigger. It also allows the analog signal to be read directly via the ADCs, which are used to estimate the threshold programmed into the Schmitt triggers (Section 3.2).

FPGA digital backend. The Lattice MachXO2-7000HE FPGA is the central processing unit of the digital system and the primary object of this thesis and receives the 60 binary event lines from the Schmitt trigger comparators, manages the complete digital processing chain computing automated ADC/DAC calibration, parallel event capture and timestamping across all 60 channels, lossless ping-pong buffering, packet assembly, and high-speed streaming to the host PC. The FPGA is clocked by an external 50 MHz crystal oscillator and communicates with the host via a UART serial link at 2 Mbit/s, converted to USB by an on-board FTDI bridge. The hardware architecture is described in detail in Chapter 4.

Host computer and GUI. A MATLAB-based graphical user interface (GUI) running on the host PC manages the full acquisition workflow, initiating and supervising the calibration sequence, computing per-channel detection thresholds from the raw ADC data, and transmitting the resulting threshold values back to the FPGA. During acquisition, it decodes the incoming binary event stream in real time, applies configurable post-processing, logs timestamped spike events to disk, and displays the network’s spatiotemporal activity as a scrolling logic-analyzer raster plot. The host software is described in Chapter 5.

3.1.2 Signal path

Extracellular action potentials, with typical peak-to-peak amplitudes of tens to hundreds of microvolts, are captured by the MEA electrodes and routed to the corresponding AFE boards, where they are amplified and band-pass filtered to yield conditioned signals at the AFE output, whose output is directly driven to the motherboard, where each conditioned signal is scaled by the INA and presented to the Schmitt trigger comparator. Every threshold crossing generates a rising edge on the corresponding FPGA input pin.

Inside the FPGA, the 60 binary input lines are continuously monitored in parallel thanks to a per-channel synchronizer chain that filters metastability and glitch noise, and a rising-edge detector that identifies valid threshold-crossing events (`edge_detector`, Section 4.5). Detected events are accumulated within fixed-length sampling windows and assembled into timestamped data packets, as described in Section 3.3, achieving an average output data rate of 12 kB/s — a reduction exceeding 99% relative to conventional continuous sampling.

3.1.3 Operational modes

Within each acquisition session, the system operates in two sequential modes, both managed through the MATLAB GUI.

Calibration mode. Before data acquisition begins, the FPGA executes a supervised calibration routine (`calibration_control`, Section 4.3) to determine the per-channel detection thresholds. It sequentially acquires 1000 background-noise samples from each of the 60 ADC channels at a sampling rate of 10 kHz (corresponding to a 100 ms recording window per channel) and streams the raw 12-bit values to the host PC over the same UART link used for event streaming. The GUI processes the received data, estimates the per-channel noise level using a hybrid algorithm based on the Quiroga MAD estimator [32], and sends 60 threshold bytes back to the FPGA that programs these values into the DACs, completing the configuration of the 60 Schmitt trigger reference voltages in approximately 9 s.

Alternatively, if the recording conditions have not changed since the previous session, the calibration phase can be skipped via a dedicated GUI command. In this case, the FPGA retains its current DAC configuration and transitions directly to acquisition mode.

Acquisition mode. After calibration, the GUI transmits a three-byte start command that includes the desired sampling period T_s so that the FPGA begins a continuous acquisition of events while operating simultaneously on all 60 channels. Every T_s , a 96-bit packet is generated and transmitted via UART, following the procedures described in Section 4.6.

The continuous acquisition continues until the user issues a stop command from the GUI, which resets the FPGA implementation state and terminates the acquisition on the host side as well.

3.2 Hardware Platform

This section provides the electrical characterization necessary to understand the hardware design constraints. In fact, the hardware platform underlying the FPGA digital back-end comprises four physical subsystems that were designed and validated as part of the prior MCU-based implementation [42] and have been unchanged: the MEA interface, the per-channel analog front-end, the motherboard, and the host communication bridge. A simplified view of the single-channel signal path is shown in Figure 3.2, whilst Figure 3.3 shows the hardware platform.

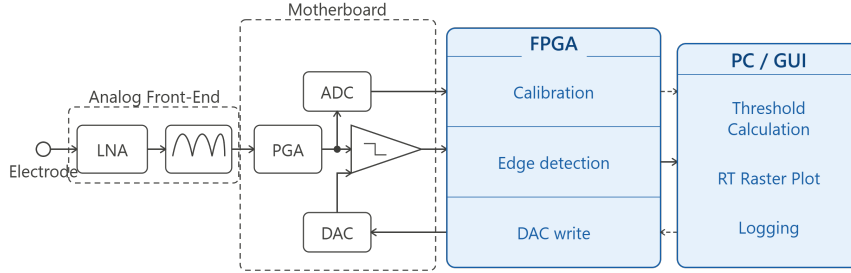


Figure 3.2. **Single-channel signal path.** From electrode to FPGA input pin: the extracellular signal is amplified and band-pass filtered by the AFE, scaled by the INA, and converted to a digital event by the per-channel Schmitt trigger comparator whose reference voltage is set by the DAC. The FPGA and PC/GUI blocks, highlighted in blue, represent the contributions of this thesis. Adapted from [42].

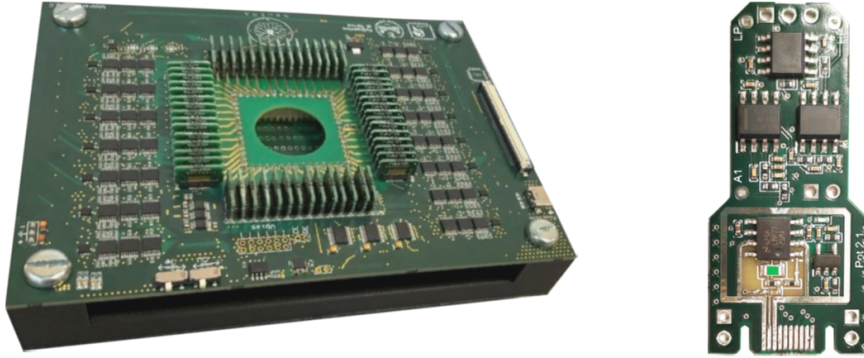


Figure 3.3. **Platform photographs.** The complete multichannel event-driven platform and a single analog front-end (AFE) printed circuit board. Source: [42].

3.2.1 MEA interface and analog front-end

The platform interfaces with standard passive MEAs of up to 60 channels, compatible with the Multichannel Systems (MCS) pad layout, through spring-loaded contacts, providing low-resistance electrical access to up to 60 recording sites and the common reference electrode.

Each of the 60 signal lines is routed to a dedicated, vertically mounted AFE printed circuit board. The AFE performs two signal conditioning functions in cascade. First, a low-noise amplifier (LNA) amplifies the extracellular signal across multiple stages, reaching a total voltage gain of 2000 V/V. Second, a band-pass filter with cut-off frequencies at 100 Hz and 5 kHz removes DC offsets and power-line interference at the low end, and electromagnetic noise and aliasing components at the high end, while preserving the spectral content of extracellular action potentials. The modular, plug-in form factor of the AFE boards allows individual channels to be replaced or upgraded without altering the motherboard or the digital backend.

3.2.2 Motherboard and event-detection chain

The motherboard is the central hardware unit of the platform, whose task is to provide regulated power to all modules, regulate the outputs of the AFE, and implement the event detection circuitry for each individual channel.

Instrumentation amplifier (INA). Each conditioned AFE output is individually routed through an INA, which scales the signal by a factor of 5 and shifts it by 1.6 V to centre it within the dynamic range of the subsequent stages.

Schmitt trigger comparators. Spike detection is implemented in hardware by 60 independent Schmitt trigger comparators with hysteresis, one per channel, which monitor

the INA output. To capture both lobes of the biphasic spike waveform, each channel compares the conditioned signal and its inverted replica against the per-channel reference voltage and combines the two branches through a logical OR, so that a deflection of either polarity beyond the threshold asserts a logic-high pulse. These 60 binary event lines are then connected to FPGA input pins, where they are processed in parallel by the edge-detection module (`edge_detector`, Section 4.5). The reference voltages are individually programmed by the FPGA during the calibration phase (Section 4.3).

Analog-to-digital converters. Baseline noise acquisition during calibration is performed by eight Texas Instruments ADS7038-Q1 12-bit successive-approximation ADCs, each providing eight single-ended input channels, for a total of 64 available channels (60 of which are connected to electrode signals). The ADCs share a common 3.3 V analog supply ($V_{DD} = 3.3$ V), yielding a full-scale input range of 0 to 3.3 V with a voltage resolution of $3.3/4096 \approx 0.8$ mV per LSB. Each device communicates with the FPGA over SPI operating in Mode 0 (CPOL = 0, CPHA = 0) at 1 MHz.

Digital-to-analog converters. The Schmitt trigger reference voltages are set by five Analog Devices AD8802/AD8804 8-bit DACs, each providing twelve independently programmable output channels, for a total of 60 available channels. Like the ADCs, these devices are powered from a 3.3 V supply, yielding a full-scale output range of 0 to 3.3 V with a voltage resolution of $3.3/256 \approx 12.9$ mV per LSB. Each device communicates with the FPGA over SPI operating in Mode 3 (CPOL = 1, CPHA = 1) at 1 MHz.

3.3 Event-Driven Acquisition Paradigm

Conventional MEA acquisition systems digitize the raw extracellular signal on every channel continuously, at sampling rates between 10 and 50 kS/s per channel. While this approach preserves full waveform morphology, it generates data volumes that scale linearly with channel count, recording duration, and sampling rate, imposing heavy demands on storage, transmission bandwidth, and real-time processing. For a 60-channel system at 10 kS/s with 12-bit resolution, the raw data rate is:

$$R_{\text{cont}} = N_{\text{ch}} \times f_s \times b = 60 \times 10,000 \text{ S/s} \times 2 \text{ B} = 1200 \text{ kB/s} \quad (3.1)$$

where $N_{\text{ch}} = 60$ is the number of channels, f_s the sampling frequency, and $b = 2$ B the storage per sample (12-bit rounded up to 16-bit for byte alignment). Sustained at this rate over a typical one-hour recording session, the system would produce more than 4 GB of raw data.

For a broad class of in vitro network-level experiments — including burst detection, synchrony quantification, firing rate estimation, and real-time closed-loop stimulation — the waveform morphology of individual spikes is not the primary variable of interest. What matters is the *spatiotemporal activation map*: which electrode detected an action potential and when. The proposed platform exploits this observation by adopting a **hardware-level event-driven architecture**, in which threshold crossings are detected in analog hardware and forwarded to the FPGA as binary logic pulses, discarding the raw waveform entirely.

The FPGA then encodes these pulses as discrete timestamped events rather than digitized samples.

The primary trade-off of this approach is the irreversible loss of spike waveform information, making it unsuitable for experiments requiring morphology-based spike sorting or detailed single-unit characterization. Within the intended application domain, however, this is an acceptable cost for the substantial reduction in data rate and system complexity it enables.

3.3.1 Sampling window and sticky accumulation

The FPGA divides the acquisition timeline into fixed-length **sampling windows** of duration T_s , configurable by the host at session start via a 16-bit counter reload value transmitted over UART. The default value is $T_s = 1$ ms.

Within each window, every channel’s binary event line is monitored for rising edges, which are OR-accumulated into a 60-bit *sticky mask* recording, for each channel, whether at least one event occurred during the window; the complete implementation is described in Section 4.5. At the end of every window, the sticky mask is latched into the output data word and reset for the next window (Section 4.2.4).

An important consequence of this encoding is that the temporal resolution of event detection is limited to one event per channel per T_s window: two spikes on the same channel within T_s are indistinguishable from a single spike. At the default $T_s = 1$ ms, this constraint is benign up to firing rates of approximately 1000 Hz — well above the physiological maximum of a single neuron (~ 500 Hz, limited by the 1–2 ms absolute refractory period [8]) and above the system’s 2 ms hardware refractory period.

3.3.2 Packet format

The FPGA, for each time window, assembles a **96-bit data word** to be sent to the host PC, combining a 32-bit timestamp with microsecond resolution, the 60-bit sticky mask that encodes whether an event occurred during the time window, and a 4-bit parity field which is currently unused:

$$\text{packet} = \underbrace{\text{timestamp_cnt}}_{32 \text{ bit}} \parallel \underbrace{\text{sticky_mask}}_{60 \text{ bit}} \parallel \underbrace{\text{parity}}_{4 \text{ bit}} \quad (3.2)$$

The full bit-level definition of each field, including timestamp overflow handling and the parity scheme, is given in Section 4.2.

3.3.3 Data-rate analysis

The FPGA serializes the 96-bit data word into $96/8 = 12$ bytes, which are transmitted over UART using the `uart_fsm` serializer module (Section 4.7). Rather than transmitting each packet individually at the end of every T_s , the FPGA accumulates $N = 500$ consecutive packets in a ping-pong double buffer (`WR_module`, Section 4.6) and then streams the complete buffer of $N \times 12 = 6000$ bytes as a single burst. With the default $T_s = 1$ ms, one

burst is produced every $N \times T_s = 500$ ms, yielding an **average output data rate** of:

$$R_{\text{event}} = \frac{N \times 12 \text{ B}}{N \times T_s} = \frac{12 \text{ B}}{T_s} = \frac{12 \text{ B}}{10^{-3} \text{ s}} = 12 \text{ kB/s} \quad (3.3)$$

The data-rate reduction relative to continuous 12-bit acquisition at 10 kS/s is therefore:

$$\eta = 1 - \frac{R_{\text{event}}}{R_{\text{cont}}} = 1 - \frac{12}{1200} = 99\% \quad (3.4)$$

3.4 Development Environment and Design Constraints

This section briefly describes the target programmable device and development environment, and summarises the FPGA’s input/output interface.

3.4.1 Target device and toolchain

The digital backend is implemented on a Lattice MachXO2-7000HE FPGA, which belongs to the MachXO2 family of non-volatile programmable logic devices, whereby the configuration bitstream is stored in on-chip Flash memory and is loaded automatically at power-up without the aid of an external configuration memory.

It also features on-chip Block RAM tiles, which are used to manage the data to be transmitted, and dedicated hardware for SPI and I²C controllers (Enhanced Function Blocks, EFB). Even though the EFB hardware is present, the project implements a dedicated SPI module for communication with the motherboard (Section 4.4). The use of the dedicated module ensures precise operation in dual mode (SPI Mode 0 for the ADCs and SPI Mode 3 for the DACs), as well as ease of customisation, which the EFB does not guarantee.

All modules are written in **SystemVerilog**; synthesis, placement, routing, and bitstream generation are performed with **Lattice Diamond** (version 3.14). Functional verification during development was carried out using **ModelSim** behavioural simulation, with dedicated testbenches exercising the calibration FSM, the SPI master, and the UART stack in isolation before integration.

The FPGA is clocked by an external 50 MHz crystal oscillator (CB3LV-3C, X2 breakout footprint) connected to a dedicated clock input pin, replacing the internal RC oscillator. Furthermore, the reset strategy is dual-source, managed both at power-up, through a forced global reset, and via software, through a dedicated signal; the reset can also be forced mechanically through a physical push-button. The complete mechanism is described in Section 4.2.

3.4.2 FPGA interface summary

Table 3.1 summarises the FPGA pin groups and their function.

Signal group	Direction	Pins	Function
Event input lines	Input	60	Schmitt trigger comparator outputs
SPI MOSI	Output	1	Shared SPI data out (ADC and DAC)
SPI MISO	Input	1	Shared SPI data in (ADC only)
SPI CLK	Output	1	Shared SPI clock
ADC chip selects	Output	8	One per ADS7038-Q1
DAC chip selects	Output	5	One per AD8802/AD8804
DAC SHDN / RS	Output	2	Static high (device enabled)
UART TX	Output	1	Data to host
UART RX	Input	1	Commands from host
UART CTS	Input	1	Hardware flow control
External clock	Input	1	50 MHz oscillator
Reset button	Input	1	Active-low manual reset
LED indicators (unused)	Output	8	Status (reset, calibration, data)
Total		91	

Table 3.1. Summary of FPGA signal groups and pin counts. All signals are single-ended, 3.3 V LVCMOS.

Chapter 4

FPGA Implementation

4.1 Hardware Architecture and Module Hierarchy

The architectural description for the FPGA implementation is written entirely in SystemVerilog and is structured around a single top-level module, `main_module`, which instantiates and interconnects eight functional sub-modules (Figure 4.1, Table 4.1).

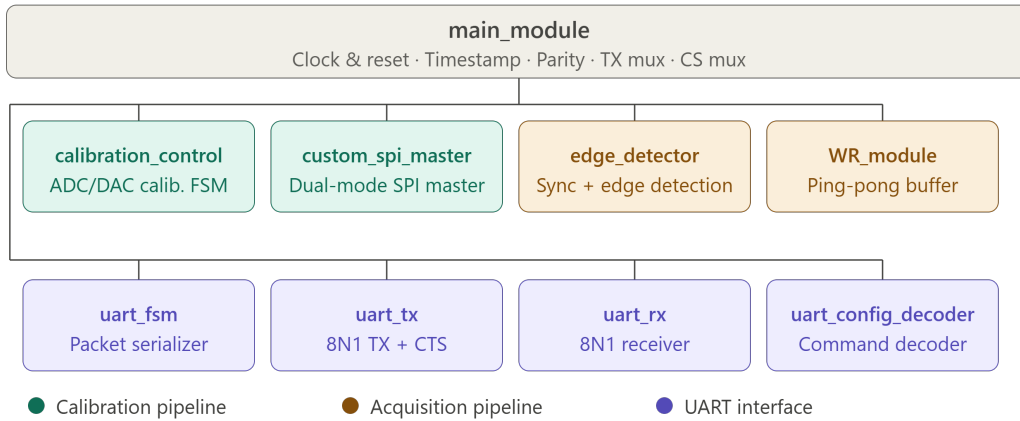


Figure 4.1. **Hardware architecture module hierarchy.** `main_module` is the top-level wrapper. Green modules form the calibration pipeline; yellow modules form the acquisition and datalogger pipeline; purple modules implement the UART interface stack.

Module	Group	Function
<code>calibration_control</code>	Calibration	32-state FSM managing ADC noise-acquisition and DAC threshold-programming pipeline.
<code>custom_spi_master</code>	Calibration	Dual-mode SPI master (Mode 0 for ADCs, Mode 3 for DACs) shared by <code>calibration_control</code> .
<code>edge_detector</code>	Acquisition	Per-channel 5-stage synchroniser, majority-vote filter, rising-edge detector, and sticky-mask accumulator.
<code>WR_module</code>	Acquisition	Ping-pong double buffer in Block RAM; accumulates $N = 500$ data words then signals <code>uart_fsm</code> .
<code>uart_fsm</code>	UART	Serialises each 96-bit data word into 12 consecutive UART bytes, MSB-first.
<code>uart_tx</code>	UART	8N1 UART transmitter with CTS hardware flow control.
<code>uart_rx</code>	UART	8N1 UART receiver with mid-bit sampling.
<code>uart_config_decoder</code>	UART	Decodes the three-byte start command and the six-zero stop command from the host.

Table 4.1. Summary of the eight sub-modules instantiated by `main_module`. Detailed descriptions are given in Sections 4.3–4.7.

4.1.1 Operational modes and signal multiplexing

The FPGA hardware operates in two mutually exclusive modes, calibration and acquisition, controlled by the `calib_active` flag asserted by `calibration_control`.

During the **calibration mode**, `calibration_control` drives the SPI master to read the ADC chips and program the DAC chips. In order to correctly control the signal path and ensure that one device is accessed before the other, two multiplexers in `main_module` regulate access to the shared resources along the calibration path. Specifically, the TX mux routes the byte stream generated by `calibration_control` to `uart_tx`, while the CS mux manages the appropriate chip-select line for either the ADC or the DAC according to the `cs_target_o` and `chip_index_o` outputs of `calibration_control`.

During **acquisition mode**, the TX mux switches to `uart_fsm`, which serializes the ping-pong buffer packets produced by `WR_module`. During acquisition mode, the 60 binary event lines from the Schmitt trigger comparators are continuously monitored in parallel by `edge_detector`, which accumulates threshold-crossing events into the sticky mask fed to `main_module` for packet assembly.

The `uart_rx` module operates during calibration and acquisition. During calibration, it forwards incoming threshold values to `calibration_control`, whilst during acquisition

only bytes intended for `uart_config_decoder` (start and stop commands) are processed.

Figure 4.2 provides a unified view of the complete session workflow, tracing the sequence from power-on and session setup through the calibration branch to the continuous event-acquisition loop and final stop condition. The modules that implement each phase are described in this chapter, whilst the software that manages the whole acquisition from threshold estimation to real-time display is described in Chapter 5.

4.2 Top-Level Module: `main_module`

`main_module` is the top-level SystemVerilog entity of the implemented hardware; it orchestrates the eight sub-modules described in Section 4.1 and implements four pieces of auxiliary logic: the dual-source reset sequencer, the microsecond timestamp counter, the 96-bit packet assembler, and the TX/CS multiplexers that arbitrate the shared UART transmitter and SPI bus between calibration and acquisition modes.

4.2.1 Dual-source reset

System reset is generated by the logical AND of two independent sources:

$$\text{general_reset_n} = \text{auto_rst_n} \wedge \text{UART_reset_n} \quad (4.1)$$

Power-on reset (`auto_rst_n`). A 26-bit saturating counter is incremented on every rising clock edge and it asserts `auto_rst_n` only after reaching 50×10^6 counts, corresponding to 1 s at the 50 MHz system clock, guaranteeing that all power rails have settled. Power-on reset can be re-triggered when an active-low push-button on the motherboard is pushed.

Datalogger enable (`UART_reset_n`). The second reset source is driven by `uart_config_decoder` (Section 4.7): it remains low until the host PC sends a valid start command, at which point it is de-asserted to release the datalogger so that event acquisition cannot begin before a correct session configuration has been received, preventing the datalogger from buffering and transmitting meaningless data.

4.2.2 Timestamp counter

Inside the module, there is a 32-bit binary counter, `timestamp_cnt`, with a temporal resolution of $1\mu\text{s}$ that wraps around after $2^{32} - 1 \approx 71.6\text{min}$ and which is updated through a 6-bit prescaler that generates the update signal and provides the timing reference for the 60 channels. If measurements exceeding the maximum counter duration are required, the host software is able to detect when the counter reaches its maximum value and add an offset of $2^{32}\mu\text{s}$, both during visualization and when saving the data.

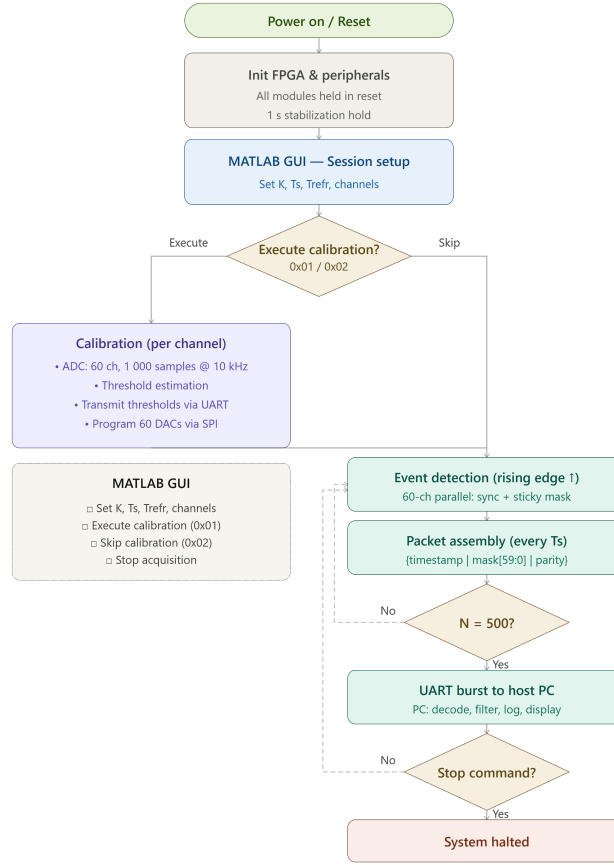


Figure 4.2. **Operational flowchart of the NeuERA acquisition platform.** The flowchart provides a high-level description of the operation of the entire project. The process begins with the initialization of the FPGA parameters and the setting of the acquisition conditions via the GUI, followed by the reset and system initialization phase. Subsequently, the flow splits into two possible paths: performing calibration via command 0x01 before proceeding with data acquisition, or skipping this phase and moving directly to event detection. If command 0x01 is sent, calibration is performed first, comprising the transmission of data from the device to the FPGA via SPI and the management and computation of the thresholds carried out on the GUI side. Only at the end of the calibration session, the flow proceeds to the datalogging phase in which event detection is handled on the FPGA side and the acquired data is stored by means of a ping-pong memory. The transmission of the first data to the host PC via UART occurs when one of the two memory banks becomes full, that is, after at least 500 packets have been stored. The FPGA hardware components are described in detail in Chapter 4, while the host-side logic is discussed in Chapter 5.

4.2.3 Packet assembly and parity

At the end of each sampling window, indicated by the `ts_tick_o` pulse generated by `WR_module`, `main_module` assembles the 96-bit output word `generated_data` by concatenating the 32-bit timestamp, the 60-bit sticky mask, and four parity bits computed combinatorially as XOR reductions over successive groups of 15 bits of the mask. The word is provided as input to `WR_module` in the same clock cycle in which `ts_tick_o` is asserted, ensuring that the timestamp, the channel bitmask, and the parity bits are always captured together as a coherent and glitch-free snapshot.

$$\text{generated_data} = \{ \text{timestamp_cnt}_{32}, \text{sticky_mask}_{60}, \text{parity}_4 \} \quad (4.2)$$

4.2.4 Sticky mask reset

After assembling and capturing `generated_data`, the sticky mask must be cleared so that the next sampling window starts from an all-zero baseline. In order to perform it, the hardware architecture implements a one-cycle delayed reset, using a single flip-flop in `main_module`. The resulting three-cycle sequence is shown in Figure 4.3.

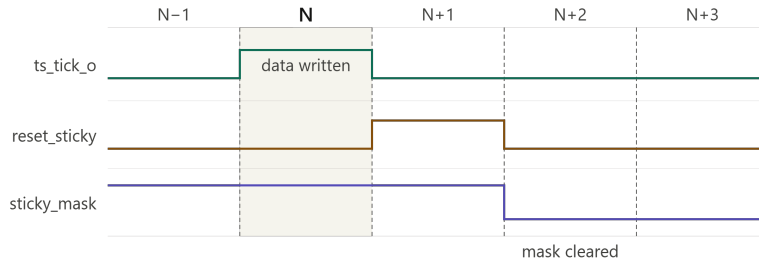


Figure 4.3. **Sticky reset timing.** `ts_tick_o` is asserted at cycle N, at the same time that `generated_data` is written into the `WR_module` buffer. In contrast, `reset_sticky` is asserted one clock cycle later, at N+1. To provide a safety margin after data acquisition, the `sticky_mask` is cleared at N+2.

4.2.5 TX and CS multiplexers

TX multiplexer. The single `uart_tx` instance is shared between the calibration pipeline (`calibration_control`) and the acquisition pipeline (`uart_fsm`). A pair of combinatorial multiplexers in `main_module` selects the active source based on the `calib_active` flag asserted by `calibration_control`. Since `calib_active` and acquisition mode are mutually exclusive by design, the flag transitions only when `calibration_control` has completed its FSM sequence and all outstanding UART bytes have been transmitted.

CS multiplexer. The SPI bus is used by two different device families, both employed during calibration mode: the eight ADS7038-Q1 ADC chips and the five AD8802/AD8804 DAC chips, each with a dedicated active-low chip-select line. During calibration, it is

therefore necessary that the bus be used by the right peripheral at the right time. The `calibration_control` drives two outputs: `cs_target_o`, which is 0 when targeting the ADCs and 1 when commanding the DACs, and `chip_index_o`, which `main_module` decodes to assert exactly one chip-select at a time, while all remaining chip-selects are held high.

4.3 Calibration Pipeline: `calibration_control`

The `calibration_control` module implements the fully automated hardware calibration pipeline as a single synchronous FSM comprising 32 states, driving the shared SPI bus and the UART transmitter through two sequential active phases: ADC noise acquisition and DAC threshold programming, with a passive reception phase in between. The complete state-transition graph exported from Lattice Diamond is shown in Figure 4.4, whilst a high-level view of the five principal phases is shown in Figure 4.5.

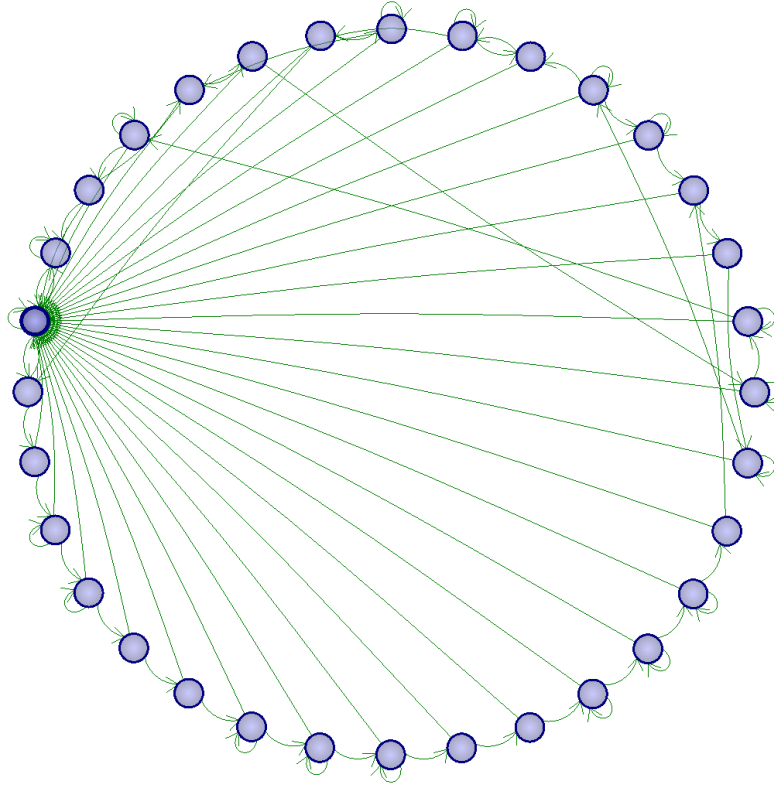


Figure 4.4. **Complete calibration FSM** as exported from Lattice Diamond (32 states). The highly connected node on the left is the IDLE state, from which all abort paths return.

4.3.1 Control protocol

From the IDLE state, the FSM waits for a single-byte command received from the host over UART:

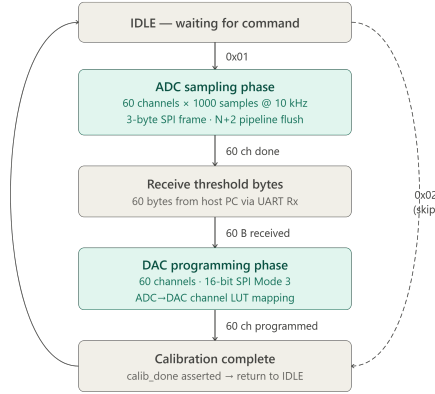


Figure 4.5. **High-level calibration FSM phases.** Command 0x01 triggers the full pipeline; command 0x02 (dashed path) bypasses the active phases and retains the existing DAC configuration.

- **0x01 — Start calibration.** The FSM transitions to the ADC phase and executes the full pipeline for all 60 channels.
- **0x02 — Skip calibration.** The FSM transitions directly to DONE, bypassing ADC sampling and DAC programming entirely. The DAC registers retain their values from the previous calibration session.

An **abort mechanism** is active in all non-idle states: if `datalogger_en_i` is deasserted while the FSM is executing any active phase, the FSM returns unconditionally to IDLE, resetting all internal counters. This allows the host to cancel a calibration in progress without a hardware reset.

The flag `calib_active` is asserted for the entire duration of the calibration pipeline (from the first transition out of IDLE until the DONE state is reached) and is used by `main_module` to hold the TX and CS multiplexers in the calibration configuration and by `uart_config_decoder` to gate incoming UART bytes, ensuring that start/stop commands from the host are not misinterpreted as threshold values during the reception phase.

4.3.2 Channel addressing scheme

The 60 electrode channels are distributed across eight ADS7038-Q1 ADCs, each exposing eight single-ended input channels (for a total of 64 available inputs; the last four of the eighth chip are unused). The FSM maintains a global channel counter `ch_cnt` in the range $[0, 59]$, from which the chip index and the intra-chip channel index are derived:

$$\text{chip_index} = \lfloor \text{ch_cnt} / 8 \rfloor \quad (4.3)$$

$$\text{intra_ch} = \text{ch_cnt} \bmod 8 \quad (4.4)$$

The chip index drives the CS multiplexer in `main_module`, which asserts the corresponding `spi_cs_adc_n[chip_index]` line, while the intra-chip channel index is written into the `MANUAL_CH_SEL` register of the selected ADC chip.

4.3.3 ADC sampling phase

For each of the 60 channels, the ADC sampling phase executes the following procedure.

Channel selection

The FSM initiates a 24-bit (3-byte) SPI write transaction to the `MANUAL_CH_SEL` register of the target ADS7038-Q1 chip at register address `0x11`. The transaction frame is structured as:

$$\text{CH_SEL frame} = \underbrace{0x08}_{\text{WR_REG opcode}} \parallel \underbrace{0x11}_{\text{register addr}} \parallel \underbrace{\text{intra_ch}[2:0]}_{\text{ch. index}} \quad (4.5)$$

The FSM waits for `spi_busy_i` to de-assert before proceeding, ensuring the transaction is complete before any subsequent operation on the same bus.

N+2 pipeline flush

The ADS7038-Q1 employs a two-stage internal acquisition pipeline, which means that once a manual channel selection command has been issued, the next two samples must be discarded as they still refer to the previously selected channel. To comply with the ADC's acquisition logic, the FSM performs two additional 24-bit SPI read transactions after writing the channel selection, discards all received bytes, and from that point onward begins the actual acquisition of the desired signal.

Sample acquisition loop

Upon completion of the pipeline flush, the FSM executes a 1000-cycle inner loop, sequentially traversing the following sub-states:

1. **Wait_10kHz.** A 13-bit timer counts 3769 system clock cycles (75.4 μ s); the remaining $\approx 24 \mu$ s of the 100 μ s inter-sample interval are consumed by the SPI transaction (3 bytes at 1 MHz).
2. **Start_SPI.** The FSM asserts `spi_start_o` with a 24-bit read frame (three dummy zero bytes) to clock out the next conversion result from the ADC MISO line.
3. **Acquire_Bytes.** As the SPI transaction progresses, each byte signalled by `spi_valid_i` is processed in order: the first byte (carrying ADC bits [11:4]) and the second byte (carrying ADC bits [3:0] in its upper nibble) are written sequentially into the internal `cal_ram` buffer at the current pointer `ptr`; the third byte is discarded as it belongs to the preamble of the next SPI frame.

4. **Next_Sample.** Once `spi_busy_i` de-asserts, the sample counter is incremented: if fewer than 1000 samples have been collected, the loop returns to `WAIT_10kHz`; otherwise the FSM exits the acquisition loop and advances to the streaming phase described below.

The ADS7038-Q1 returns the 12-bit conversion result in the first two bytes of the response frame, as detailed in Table 4.2. The host reconstructs the full 12-bit value as:

$$\text{val}_{12} = (\text{byte}_0 \ll 4) \mid (\text{byte}_1 \gg 4) \quad (4.6)$$

Byte	Bit field	Content
0	[7:0]	ADC result bits [11:4] (MSBs)
1	[7:4]	ADC result bits [3:0] (LSBs)
1	[3:0]	Zero-padded
2	[7:0]	Preamble of next frame — discarded

Table 4.2. ADS7038-Q1 SPI response frame layout in manual channel-select mode (24-bit transaction).

After 1000 samples have been acquired and transmitted for one channel, the FSM increments `ch_cnt`. If `ch_cnt` < 60, the channel-select procedure described above is repeated for the next channel, using the updated `chip_index` and `intra_ch` values. Otherwise, if `ch_cnt` reaches 60, all channels have been processed and the FSM transitions to the threshold reception phase.

At the end of the entire ADC data acquisition process, the FPGA has transmitted $60 \times 1000 \times 2 = 120,000$ bytes to the host, in a time approximated as follows (without considering the channel selection write and the two dummy reads):

$$t_{\text{ADC}} \approx 60 \times 1000 \times 100 \mu\text{s} = 6 \text{ s} \quad (4.7)$$

Taking into account the threshold reception phase and the threshold comparator programming, the total calibration time rises to about 9s.

4.3.4 Threshold reception

Following the ADC phase, the FSM transitions to the `RECEIVE THRESHOLDS` state and waits for the host PC to transmit exactly 60 bytes, each providing the 8-bit DAC code for a single channel in sequential order from channel 0 to 59. The incoming bytes are processed via `uart_rx`: whenever a valid byte is detected (`rx_valid_i` = 1), the FSM writes it to a 60-element array at the current `th_cnt` index and increments the counter. Since there is no timeout condition, the machine will stall in this state indefinitely until the full payload arrives or an abort is issued.

4.3.5 DAC programming phase

SPI frame format

Each DAC write transaction transfers a 16-bit frame to one channel of one AD8802/AD8804 chip via the shared SPI bus (Mode 3, 1 MHz):

$$\text{DAC frame} = \underbrace{0000}_{[15:12], \text{ d.c.}} \parallel \underbrace{A_3 A_2 A_1 A_0}_{[11:8], \text{ ch. addr}} \parallel \underbrace{D_7 \dots D_0}_{[7:0], \text{ threshold}} \quad (4.8)$$

The FSM asserts `spi_start_o` with `cpol_i` = 1 (Mode 3), waits for `spi_busy_i` to de-assert, then increments the DAC channel counter and repeats until all 60 channels have been programmed.

The hardware implementation, since the physical wiring on the motherboard PCB does not map ADC channel i to DAC channel i , encodes the correct correspondence as a 60-entry hardcoded look-up table applied combinatorially, assigning each global ADC channel index a target DAC chip index (0–4) and a DAC intra-chip channel address (0–11), so that the threshold computed for ADC channel i is always written to the comparator whose reference voltage controls exactly that electrode.

4.3.6 SPI mode switching

The ADC and DAC peripherals require different SPI clock polarities: Mode 0 (CPOL = 0) for the ADS7038-Q1 and Mode 3 (CPOL = 1) for the AD8802/AD8804. The FSM selects the active mode by driving the `cpol_i` input of the shared `custom_spi_master` instance: `cpol_i` = 0 for all ADC transactions and `cpol_i` = 1 for all DAC transactions. The SPI master incorporates a dedicated `CS_SETUP` state (Section 4.4) that brings the SCLK line to the correct idle level before asserting chip-select, preventing the spurious half-clock-period glitch that would otherwise appear at the transition between Mode 0 and Mode 3.

4.4 SPI Master: `custom_spi_master`

The `custom_spi_master` module provides a single, shared SPI bus master that serves both the ADS7038-Q1 ADC family (SPI Mode 0) and the AD8802/AD8804 DAC family (SPI Mode 3). It is instantiated once in `main_module` and driven exclusively by `calibration_control` via a start/busy/valid handshake. The module's complete state-transition graph as exported from Lattice Diamond is shown in Figure 4.6.

4.4.1 Module parameters

The SPI master relies on several elaboration-time parameters to govern its timing behavior and transaction geometry:

- **CLK_FREQ** = 50 MHz: The main system clock frequency.
- **SPI_FREQ** = 1 MHz: The target frequency for the generated SPI clock.

- **Idle.** The module rests in the IDLE state with `spi_busy_o` de-asserted, signalling to `calibration_control` that the bus is free. A rising edge on `spi_start_i` triggers the state transition: the FSM latches the incoming transaction parameters (`burst_len_i`, `spi_tx_data_i`, `cpol_i`) and advances to `CS_SETUP`.
- **Cs_Setup.** This is a transitional state whose purpose is to bring the SCLK line to its idle level, ensuring compliance with the default timing requirements of both the ADCs and the DACs. Over the course of a single clock cycle, the line is given time to settle at `cpol_lat` before `CS_N` is pulled low. The SCLK idle level is low for Mode 0 ($\text{CPOL} = 0$) or high for Mode 3 ($\text{CPOL} = 1$).
- **Cs_Low.** `CS_N` is asserted and the MSB of the transmit shift register is driven onto `MOSI`. The FSM advances to `SHIFT_A` on the next clock-divider tick, making this state a pure setup cycle for the first bit.
- **Shift_A and Shift_B.** These two states represent the operational core of the FSM. Together, they correspond to the two half-periods of a single SCLK cycle and are responsible for transferring exactly one bit of data. A key feature of this module is its support for both Mode 0 and Mode 3. Both modes share the same sampling convention: data is captured (read) on the rising clock edge and updated (written) on the falling clock edge. However, since they start from opposite idle levels, the roles of `SHIFT_A` and `SHIFT_B` are swapped depending on the selected mode.
 - **Mode 0** ($\text{CPOL} = 0$): SCLK idles low. In `SHIFT_A`, the clock transitions from 0 to 1 (rising edge), at which instant the module samples the incoming bit on `MISO` and stores it in the receive shift register `rx_shift`. In `SHIFT_B`, the clock makes the opposite transition from 1 to 0 (falling edge): at this point `MOSI` is updated with the next bit to be transmitted, ensuring it is stable and ready to be read by the connected slave.
 - **Mode 3** ($\text{CPOL} = 1$): the procedure is analogous, but starting from the high idle level; the two transitions are therefore inverted, and the Mode 3 protocol is thus fully satisfied.

At the end of `SHIFT_B`, the FSM checks whether all bytes in the burst have been transmitted.

- **Byte_Done.** This state is activated upon completion of the 8-bit transfer: the received byte is saved in `rx_data_o`, `data_valid_o` is pulsed, and the byte index is incremented. If the entire transaction is complete, the clock is frozen at its idle level and the FSM transitions to `CS_HIGH`. If further bytes remain, the next byte is loaded, its MSB is pre-driven onto `MOSI` (in Mode 0), and the FSM returns to `SHIFT_A` to continue the transfer.
- **Cs_High.** This state manages the clean termination of the SPI transaction by de-asserting chip-select (`spi_cs_n_o` held high) while maintaining the clock at the idle level for the current mode. The FSM dwells in this state for `CSH_CYCLES` clock cycles to satisfy the minimum CS high time t_{CSH} required by the peripheral devices. Once the hold time elapses, `busy_o` is de-asserted and the FSM returns to IDLE.

SPI Dual-Mode Timing: Mode 0 vs Mode 3

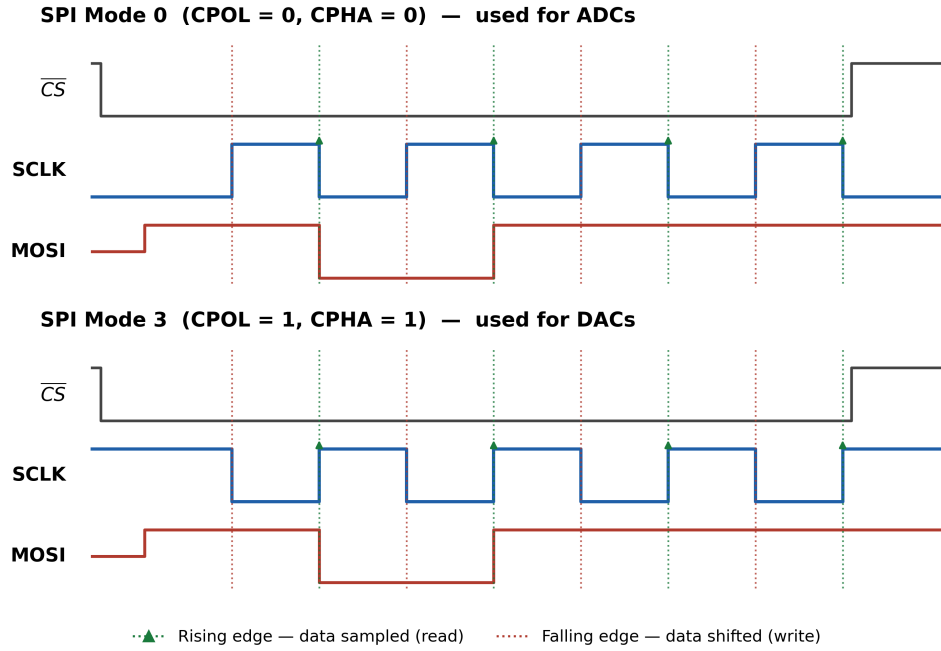


Figure 4.7. **SPI Mode 0 vs Mode 3 timing diagram.** In both modes, data is sampled on the rising edge of SCLK and shifted on the falling edge; the only difference is the clock idle level (low for Mode 0, used for the ADCs; high for Mode 3, used for the DACs).

4.4.3 Handshake interface

The SPI master exposes a three-signal handshake to `calibration_control`:

- **`spi_start_i`**: one-cycle pulse asserted by the caller to begin a transaction. Must be de-asserted by the next clock edge; multi-cycle assertion is not supported.
- **`spi_busy_o`**: asserted from the moment IDLE is left until the moment IDLE is re-entered. `calibration_control` polls this signal to detect transaction completion.
- **`spi_valid_o`**: one-cycle pulse asserted in `BYTE_DONE` when the last byte has been received. On this cycle, `spi_data_i` (the received byte) is valid and is captured by `calibration_control`.

4.5 Event Detection: `edge_detector`

The `edge_detector` module is the functional core of the acquisition pipeline: it is the component directly responsible for converting the 60 binary lines from the Schmitt trigger comparators into the timestamped event stream that defines the platform’s event-driven paradigm. The module operates entirely combinatorially and sequentially in parallel across

all 60 channels — there is no shared state between channels, and no channel can stall or delay another. The complete per-channel datapath is illustrated in Figure 4.8 and it comprises four stages, executed identically and independently for each of the $N_{\text{CH}} = 60$ channels: a synchroniser/glitch-filter shift register, a majority-vote debouncer, a rising-edge detector, and a sticky-mask OR-accumulator.

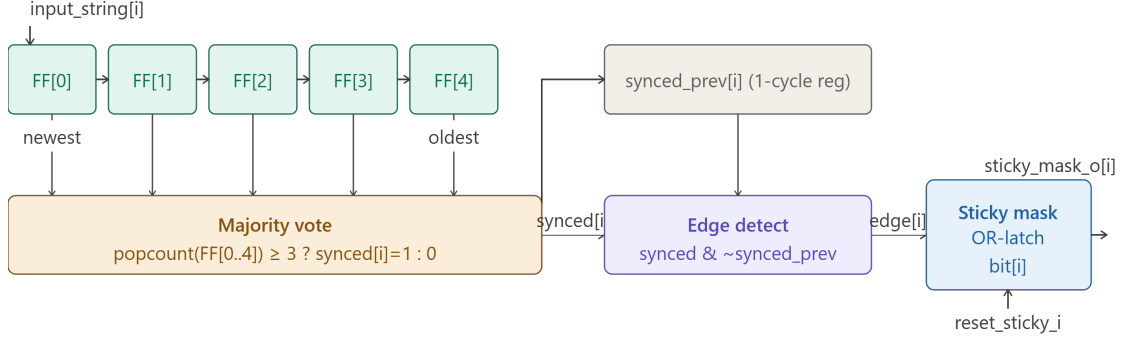


Figure 4.8. **Per-channel event detection datapath.** For each channel, a front-end 5-stage shift register composed of five interconnected flip-flops receives the raw input signal. If the majority-vote logic determines that the population count of high bits within the register is greater than or equal to `MAJ_THRESH`, `synced[i]` is asserted; this signal is then compared against a one-cycle-delayed copy of itself to verify that a rising edge has genuinely occurred. The detected edge is OR-accumulated into `sticky_mask_o[i]`, which is subsequently cleared by `reset_sticky_i`.

4.5.1 Synchroniser and glitch-filter chain

Each of the 60 raw input lines, `input_string[i]`, originates from an analog Schmitt trigger comparator on the motherboard and is therefore fully asynchronous with respect to the FPGA’s 50 MHz system clock. Sampling an asynchronous signal directly with combinatorial logic risks metastability — a transient, undefined voltage level at the flip-flop output when the input transitions arbitrarily close to a clock edge — which can propagate unpredictably through downstream logic. In order to mitigate it, every channel feeds a dedicated shift register of `CHAIN_LEN = 5` flip-flops, instantiated per-channel as a `generate` loop:

$$\text{ff_chain}[i] \leftarrow \{ \text{ff_chain}[i][\text{CHAIN_LEN} - 2 : 0], \text{input_string}[i] \} \quad (4.9)$$

At every rising clock edge, the new sample of `input_string[i]` enters position `ff_chain[i][0]` (the most recent sample), while older samples shift toward `ff_chain[i][CHAIN_LEN-1]` (the oldest, 5 cycles in the past).

4.5.2 Majority-vote debouncing

Rather than using the synchronized signal directly, the module computes a **majority vote** over the entire 5-bit shift-register window. For each channel i , the population count (number of bits equal to 1) of `ff_chain[i]` is compared against a fixed threshold:

$$\text{MAJ_THRESH} = \left\lfloor \frac{\text{CHAIN_LEN}}{2} \right\rfloor + 1 = \left\lfloor \frac{5}{2} \right\rfloor + 1 = 3 \quad (4.10)$$

$$\text{syncd}[i] = \begin{cases} 1 & \text{if } \sum_{j=0}^4 \text{ff_chain}[i][j] \geq 3 \\ 0 & \text{otherwise} \end{cases} \quad (4.11)$$

This computation is purely combinatorial and is implemented in the hardware as a nested loop that accumulates the population count across the 5-bit window and compares it against `MAJ_THRESH`.

The precise condition for `syncd[i]` to assert is that **at least 3 of the 5 most recent samples of `input_string[i]` are equal to 1**. A single isolated sample at 1 contributes a population count of 1 for exactly 5 cycles as it transits the shift register, and is therefore always rejected ($1 < 3$). However, two isolated single-cycle pulses spaced sufficiently close together — such that their respective 5-cycle "presence windows" overlap with a third pulse — can together satisfy the 3-out-of-5 condition and trigger `syncd[i]`, even if none of the individual pulses persists for more than one cycle. The filter therefore guarantees rejection of *any single isolated single-cycle glitch in complete isolation*, but does not guarantee rejection of clustered multi-pulse noise patterns. Figure 4.9 illustrates the behaviour for two representative input transients over a 15-cycle window.

Detection latency. A direct consequence of the implemented system is a fixed detection latency, since a minimum of 3 clock cycles (60 ns) must elapse from the moment the first sample of the new level enters the shift register. The resulting detection latency is therefore deterministic and identical across all channels, introducing no inter-channel skew, unlike a conventional MCU-based implementation.

4.5.3 Rising-edge detection

A second register, `syncd_prev`, captures the value of `syncd` from the previous clock cycle:

$$\text{syncd_prev}[n+1] = \text{syncd}[n] \quad (4.12)$$

A rising edge on channel i — i.e., a $0 \rightarrow 1$ transition of the debounced signal — is then detected combinatorially as:

$$\text{rising_edge_det}[i] = \text{syncd}[i] \wedge \overline{\text{syncd_prev}[i]} \quad (4.13)$$

By construction, `rising_edge_det[i]` is asserted for *exactly one clock cycle* per detected spike: on the cycle immediately following the transition of `syncd[i]` from 0 to 1, `syncd_prev[i]` still holds the previous value 0, so the AND condition is satisfied; on the subsequent cycle, `syncd_prev[i]` has been updated to 1, and the condition is no longer met (regardless of whether `syncd[i]` remains high).

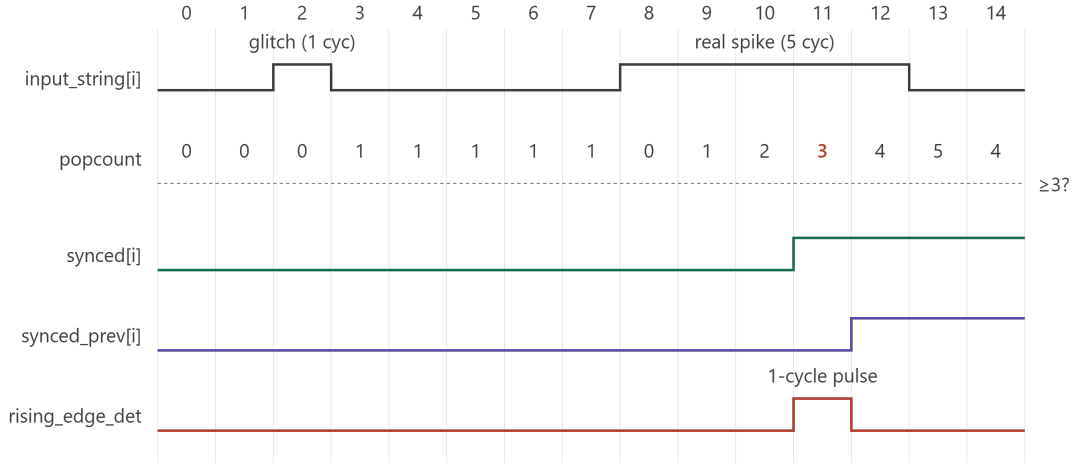


Figure 4.9. **Majority-vote debouncing — example waveform.** A 1-cycle glitch at cycle 2 raises the population count to at most 1 (cycles 3–7) and is fully rejected by cycle 8. A sustained transient beginning at cycle 8 progressively raises the population count, crossing the `MAJ_THRESH = 3` threshold at cycle 11 — three cycles after the input transition — producing a single-cycle pulse on `rising_edge_det`.

4.5.4 Sticky-mask accumulation

The final stage accumulates `rising_edge_det` into the 60-bit `sticky_mask_o` register using an OR-latch:

$$\text{st_mask_o}[n+1] = \begin{cases} 0 & \text{if } \text{reset_sticky_i}[n] = 1 \\ \text{st_mask_o}[n] \vee \text{ris_edge_det}[n] & \text{otherwise} \end{cases} \quad (4.14)$$

Once `sticky_mask_o[i]` is set to 1 by a detected edge, it remains 1 — regardless of any further edges or the absence thereof on channel i — until `reset_sticky_i` is asserted. This is the mechanism by which the module answers the question “*did channel i fire at least once during the current sampling window?*”, independent of how many times it fired.

4.6 Datalogger: `WR_module`

`WR_module` implements a zero-loss acquisition buffer based on a double-buffered (*ping-pong*) architecture, comprising two independently addressed Block RAM banks, `A_memory` and `B_memory`. The module is organised around two synchronous processes: a **write process**, driven by a free-running timer operating at the sampling period, and a **read process**, driven by the `next_data` handshake from `uart_fsm`. This architecture allows data to be read from one memory bank while the other is being written at 1 kHz, eliminating any risk of overwriting a packet before it has been transmitted, or of corrupting data prior to its dispatch via the much slower burst-mode UART transmission (Section 3.3). In summary,

while one memory bank accumulates newly generated packets, the other is read out and streamed to `uart_fsm`.

Both `A_memory` and `B_memory` are instantiated as 500×96 -bit arrays using the synthesis attribute (`* ram_style = "block" *`). Indeed, the employed FPGA allows dedicated Block RAM tiles to be employed rather than inferring distributed RAM from logic cells, thereby reducing LUT utilisation and guaranteeing single-cycle, fixed-latency read/write access regardless of buffer depth.

4.6.1 Write process

A 32-bit free-running `counter` increments on every clock cycle while `counter < counter_limit`, where `counter_limit` is assigned directly from `UART_counter_end` — the 16-bit sampling-period value received from the host during session start (Section 4.7). When `counter` reaches `counter_limit`:

1. `counter` resets to 0, restarting the period.
2. `ts_tick_o` is asserted for exactly one clock cycle, generating `reset_sticky` (Section 4.2.4).
3. The current value of `generated_data` is written to `A_memory[wr_index]` if `writing_AB = 1`, or to `B_memory[wr_index]` otherwise.
4. `wr_index` is incremented; if it has not yet reached $N - 1$ (i.e., 499), the active bank is unchanged and `check_save` does not toggle. Conversely, once $N - 1$ is reached, `wr_index` resets to 0, the selected bank is switched by inverting `writing_AB`, and a one-cycle pulse is generated on `check_save`.

4.6.2 Read process

The read process is driven by `check_save` and `next_data` from `uart_fsm`, Section 4.7.

When `check_save` toggles, the read process activates: `reading_active` is set to 1 and `rd_index` is initialised to 1, so as to begin data retrieval from the second entry. Furthermore, `output_buffer` is immediately preloaded with packet 0 of the bank that has just become inactive, ensuring that the datum is already available on the following clock cycle and that no additional latency need be introduced at the start of each burst.

This preload makes packet 0 immediately available to `uart_fsm` without requiring an initial `next_data` pulse, eliminating one cycle of latency at the start of every burst.

`next_data` advances the read pointer while `reading_active` is asserted and no new `check_save` has arrived; under this condition, `rd_index` is free to advance and packet `rd_index` is loaded into `output_buffer` to be forwarded to the UART blocks. If instead `rd_index` reaches the maximum value ($N = 500$), `reading_active` is reset to 0, thereby terminating the data burst.

With $T_s = 1$ ms set via the GUI, the complete acquisition cycle therefore requires $T_s \times N = 500$ ms to fill one full memory bank, while the streaming burst takes approximately 30 ms, as described in Section 3.3.

4.7 UART Interface Stack

The FPGA-to-PC interface is implemented over UART through four separate modules, each with a distinct function, which together realize the complete serial link between the two devices at 2Mbit/s with 8N1 framing. The primary modules forming the physical-layer interface are `uart_tx` and `uart_rx`, implementing the byte transmitter and receiver, respectively. Protocol management is delegated to the FSM `uart_fsm`, which governs the correct activation timing of the relevant control signals. The remaining module, `uart_config_decoder`, controls the correct activation timing of the control signals, serializing each 96-bit packet into a 12-byte sequence.

4.7.1 Transmitter: `uart_tx`

`uart_tx` is a parameterised 8N1 UART transmitter with hardware flow control, implemented as a 5-state FSM (Figure 4.10):

- **Idle.** When no UART communication is required, `tx_o` is held high (UART idle level); the FSM remains in this state waiting for the start signal `start_i` to be asserted. Upon assertion, `data_i` is latched into `data_reg` and `busy_o` is asserted immediately, before the `cts_i` line is examined, guaranteeing that the calling module (`uart_fsm` or `calibration_control`) cannot overwrite `data_i` on the same or following cycle, regardless of whether the transaction proceeds immediately or is deferred by flow control. The FSM then branches on `cts_i`: if `cts_i` = 0, transmission may proceed; otherwise, the FSM must wait for the line to be released by transitioning to `WAIT_CTS`.
- **Wait_Cts.** This state is entered when `cts_i` is high. Under this condition, `tx_o` remains high and `busy_o` remains asserted. To exit, the FSM must continuously monitor host availability; it can only proceed once `cts_i` returns to 0. This state guarantees data integrity, since the contents of `data_reg` cannot be overwritten for as long as the FSM remains in this condition.
- **Start.** `tx_o` is driven low for `CLKS_PER_BIT` = 25 clock cycles, transmitting the UART start bit.
- **Data.** The byte to be transmitted is sent starting from the least significant bit, following an LSB-first transmission scheme.
- **Stop.** `tx_o` is driven high for `CLKS_PER_BIT` = 25 clock cycles, transmitting the UART stop bit. Upon completion, `busy_o` is de-asserted and the FSM returns to `IDLE`.

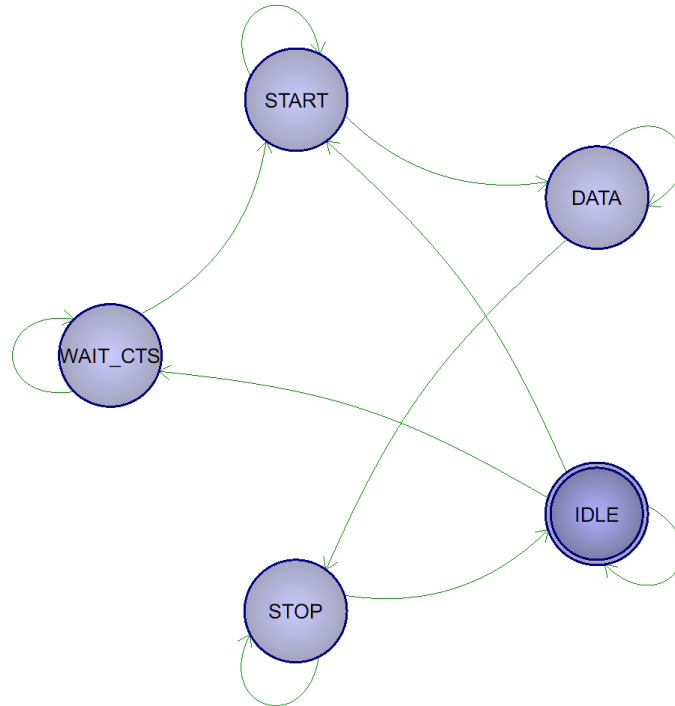


Figure 4.10. **uart_tx** FSM as exported from Lattice Diamond (5 states). `WAIT_CTS` is entered only if the CTS line is asserted at the moment `start_i` is received.

4.7.2 Receiver: **uart_rx**

uart_rx is a parameterised 8N1 receiver, where 8N1 denotes a frame format of 8 data bits, no parity bit, and 1 stop bit. The module operates at 50 MHz, shared with all other modules in the system, and receives data at a baud rate of 2 Mbaud; the number of clock cycles per bit is derived at elaboration time from these two parameters. The module is implemented as a 4-state FSM (Figure 4.11):

- **Idle.** The FSM waits for a falling edge on `rx_i` ($1 \rightarrow 0$), causing the transition to **START**.
- **Start.** Since a spurious falling edge may occur, rather than accepting the falling edge unconditionally — the receiver’s primary defence against transient line glitches being misinterpreted as the beginning of a byte — the FSM waits $\text{CLKS_PER_BIT}/2 = 12$ clock cycles and re-evaluates `rx_i`. Specifically:
 - if `rx_i` is still 0 at the mid-bit point, the start bit is confirmed; `clk_cnt` is reset and the FSM transitions to **DATA**, with subsequent samplings now aligned to the centre of each bit period, guaranteeing maximum robustness to jitter between the FPGA clock domain and the host UART;

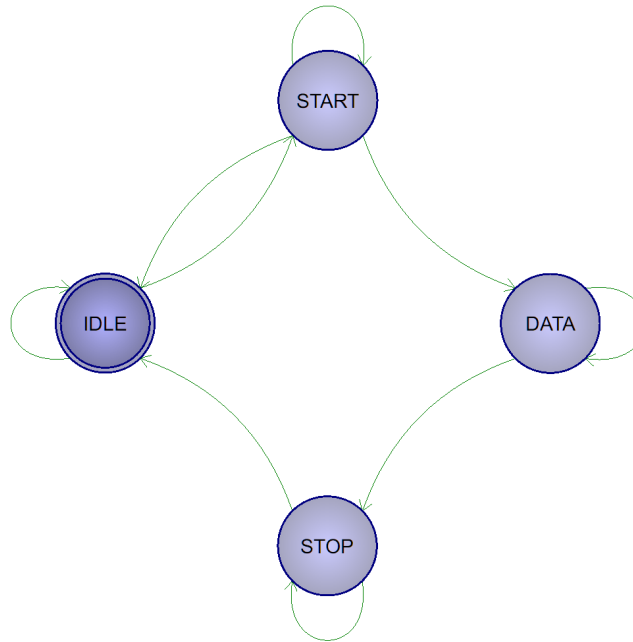


Figure 4.11. **uart_rx** FSM as exported from Lattice Diamond (4 states). The self-loop on START represents the mid-bit verification delay; a false start condition returns directly to IDLE.

- if `rx_i` has returned to 1 before the mid-bit point, the falling edge is judged spurious and the FSM returns to IDLE.
- **Data.** Once per `CLKS_PER_BIT = 25`-cycle period, `rx_i` is sampled and stored in `shift_reg[bit_cnt]`. Because sampling began at the centre of the start bit and each subsequent sample occurs exactly one full bit period later, every data-bit sample falls at the centre of its respective bit cell. Bits are received **LSB-first**, matching the transmission order of `uart_tx`. After 8 bits (`bit_cnt = 7`), the FSM transitions to STOP.
- **Stop.** The FSM waits one further `CLKS_PER_BIT`-period (the stop bit), then latches `shift_reg` into `data_o` and asserts `data_valid_o` for exactly one clock cycle before returning to IDLE. `data_valid_o` is consumed by both `calibration_control` (for threshold byte reception) and `uart_config_decoder` (for start and stop command handling).

4.7.3 Packet serialiser: `uart_fsm`

`uart_fsm` converts each 96-bit `generated_data` word supplied by `WR_module` into `NUM_BYTES = 96/8 = 12` sequential bytes transmitted via `uart_tx`. It is implemented as a 3-state FSM (Figure 4.12):

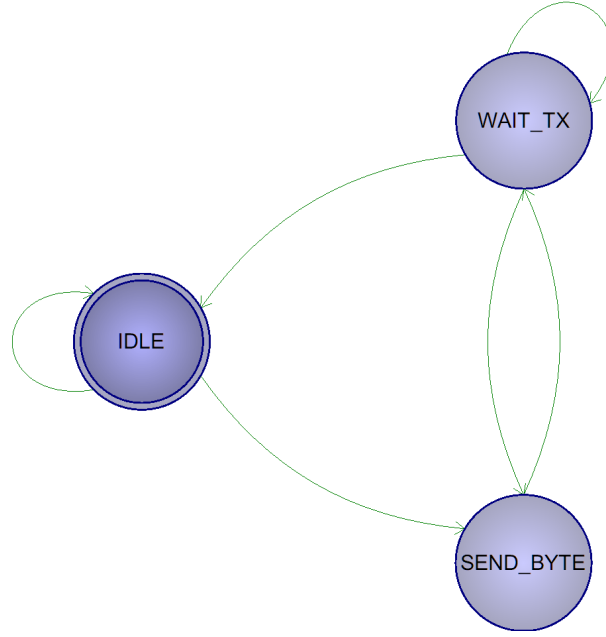


Figure 4.12. **uart_fsm** FSM as exported from Lattice Diamond (3 states). The IDLE → SEND_BYTE transition is gated by `valid_i && !next_data_o`, enforcing a one-cycle inter-packet pause.

- **Idle.** The byte counter `byte_cnt` is held at 0. The FSM transitions to SEND_BYTE only when `valid_i = 1` and `next_data_o = 0`. The second condition — that `next_data_o` (asserted for one cycle at the end of the previous packet, Section 4.6) has already returned low — enforces a mandatory one-cycle pause between consecutive packets, giving `WR_module` time to update `output_buffer` (and, if necessary, de-assert `valid_i` at a buffer boundary) before it is re-sampled.
- **Send_Byte — byte extraction order.** The byte at index `byte_cnt` is extracted from `data_i` using a descending bit-slice, MSB-first:

$$\text{uart_data_o} = \text{data_i}[(95 - 8 \cdot \text{byte_cnt}) -: 8] \quad (4.15)$$

For `byte_cnt = 0`, this extracts `data_i[95:88]` — the most-significant byte of the timestamp field. As `byte_cnt` increments from 0 to 11, successive bytes are extracted in strictly decreasing bit-significance order, so the transmitted byte sequence is:

$$\underbrace{\text{ts}[31:24], \text{ts}[23:16], \text{ts}[15:8], \text{ts}[7:0]}_{4 \text{ timestamp bytes}}, \underbrace{\text{mask}[59:52], \dots, \text{mask}[11:4]}_{7 \text{ sticky-mask bytes}}, \underbrace{\{\text{mask}[3:0], \text{parity}[3:0]\}}_{1 \text{ byte}} \quad (4.16)$$

The last byte combines the four least-significant bits of the sticky mask with the four parity bits, completing the 96-bit packet in exactly 12 bytes. `uart_start_o` is pulsed for one cycle to trigger `uart_tx`, and the FSM transitions to WAIT_TX.

- **Wait_Tx.** The FSM waits until `uart_start_o` has returned low **and** `uart_busy_i` = 0, i.e., until `uart_tx` has fully completed the byte (250 cycles, Section 4.7.1). Two outcomes follow:
 - If `byte_cnt` < 11: increment `byte_cnt` and return to `SEND_BYTE` for the next byte.
 - If `byte_cnt` = 11 (last byte just completed): assert `next_data_o` for one cycle — requesting the next packet from `WR_module` (Section 4.6) — and return to `IDLE`.

The total time to transmit one 96-bit packet is therefore $12 \times 250 = 3000$ cycles ($60\mu\text{s}$ at 50 MHz), plus the one-cycle inter-packet pause enforced in `IDLE`. Over a 500-packet burst this amounts to $500 \times 60\mu\text{s} = 30\text{ ms}$, matching the burst duration derived in Section 3.3 from the raw byte count and the UART line rate.

4.7.4 Command decoder: `uart_config_decoder`

`uart_config_decoder` is the last component of the UART block; its purpose is to parse the session-control commands issued by the host from the byte stream produced by `uart_rx`, extracting the parameters `UART_reset_n` (part of `general_reset_n`, Equation 4.1), which governs the start and stop of the entire acquisition process, and `UART_counter_end`, i.e. the sampling-period value consumed by `WR_module` (Section 4.6). The module has an input `rx_valid` which, at the top level, is defined as `decoder_rx_valid = rx_valid && !calib_active`; in this way the decoder never observes bytes intended for the calibration threshold stream.

While the system is awaiting configuration (`locked` = 0), every received non-zero byte advances `byte_index` through a three-step procedure, which operates as follows:

- In **Byte 0 (Start)**, the reception of any non-zero value drives `UART_reset_n` = 1, starting the datalogger; the module then advances to byte 1. Unlike the convention of the host GUI software, which typically sends 0x01, the hardware accepts any byte other than 0x00 as the initial trigger.
- In **Byte 1 (MSB)**, the received byte is stored in the upper half of the temporary register (`temp_counter_end[15:8]`) and the module advances to byte 2.
- In **Byte 2 (LSB and Lock)**, the byte is concatenated with the MSB to form the final 16-bit value, which is assigned to `UART_counter_end`; `byte_index` is then reset to 0 and the `locked` flag is asserted.

From this point on, with `locked` = 1, the start sequence cannot be re-triggered and any further non-zero bytes are discarded, preventing accidental reconfigurations during acquisition. The module remains in this condition until a stop command is received.

It is important to note that, since the hardware logic prioritises null bytes to handle the stop sequence, if Byte 1 or Byte 2 equals 0x00 it is intercepted by the branch that increments `zero_cnt`, preventing `byte_index` from advancing and stalling the three-byte sequence without allowing `locked` to be asserted. The protocol therefore implicitly requires that both the MSB and the LSB of `UART_counter_end` be non-zero.

Each 0x00 byte increments `zero_cnt` while any non-zero byte resets it, and once this counter reaches 2 or above, indicating at least three consecutive null bytes, the decoder stops the datalogger by de-asserting `UART_reset_n`, clearing `locked`, and resetting `byte_index` to 0, leaving the hardware ready for a new configuration.

Chapter 5

Host Software: MATLAB GUI

The complete acquisition workflow, which consists in hardware configuration, automated calibration, real-time event decoding, refractory-period filtering, data logging, and live raster visualization, is implemented in a single MATLAB script that executes sequentially through four phases: session setup and calibration, threshold computation and DAC programming, asynchronous event acquisition, and real-time display. Each phase is described in the following sections.

5.1 Session Setup and Calibration

At startup, the GUI guides the user through a sequence of modal dialogs that collect all session parameters: hardware constants (FPGA clock frequency, serial port, baud rate, front end reference voltage and gain), the acquisition parameters T_s (default 1 ms) and K_{Quiroga} , the post-processing refractory period T_{refr} (default 2 ms; set to 0 to disable), and the subset of channels to monitor via a scrollable selection table. The only chosen parameter to be sent to the FPGA via UART protocol is T_s (Section 4.7), transmitted as:

$$\text{user_counter_end} = \text{round}\left(\frac{T_s[\text{ns}] \times f_{\text{FPGA}}}{10^9}\right) - 1 \quad (5.1)$$

For the default $T_s = 1$ ms and $f_{\text{FPGA}} = 50$ MHz, $\text{user_counter_end} = 49,999 = 0\text{x}\text{C34F}$.

After the dialogs, the GUI creates a timestamped session directory, opens the serial port at 2 Mbit/s, flushes any residual bytes by transmitting the 6-byte stop sequence, and opens the log file with a tab-separated header of channel identifiers. The GUI then presents two options, one to be chosen:

- **Execute calibration (0x01):** the GUI sends the byte 0x01, triggering the calibration FSM (Section 4.3). The FPGA streams $60 \times 1000 \times 2 = 120,000$ bytes of raw ADC noise samples, which are reshaped into a 1000×60 matrix: for each channel, pairs of bytes are interpreted MSB-first and right-shifted by 4 bits to recover the 12-bit code (Section 4.3, Equation 4.6), which is then scaled to volts using the ADC's 3.3 V full-scale range. The resulting 1000×60 voltage matrix is passed to the threshold algorithm described in Section 5.2.

- **Skip calibration (0x02):** the GUI sends the byte 0x02; the FPGA bypasses ADC sampling and DAC programming and retains the threshold values from the previous session.

5.2 Threshold Algorithm

Applying the standard Quiroga MAD estimator [32] directly to the 1000-sample calibration window risks overestimating the background noise when the channel fires at high rates, since spike waveforms might contaminate the median-of-absolute-deviations. Therefore, a two-stage hybrid approach has been implemented, where a robust pre-estimate based on the 25th percentile is used to mask spike-like samples, and the Quiroga MAD is then applied to the surviving clean set.

Both stages exploit the relationship between order statistics of $|x_c|$ and σ when $x_c = x - \text{median}(x)$ is approximately Gaussian. For a probability mass p enclosed in $|z| \leq z_p$ with $z = x_c/\sigma$:

$$z_p = \Phi^{-1}\left(\frac{1+p}{2}\right) \quad (5.2)$$

giving $z_{0.25} \approx 0.3186$ and $z_{0.50} \approx 0.6745$.

Stage 1 — P25 pre-estimate and spike masking. The first step is to evaluate the rough noise using the 25th percentile of $|x_c|$, which is substantially less sensitive to high-amplitude outliers than the median, because spikes must constitute more than 75% of the window before they bias it. The calculation is performed as follows:

$$\sigma_{\text{perc}} = \frac{\text{P25}(|x_c|)}{0.3186} \quad (5.3)$$

The obtained value is used to determine whether there are any peaks within the calibration acquisition window that exceed four times the evaluated noise ($|x_c[n]| > 4\sigma_{\text{perc}}$) flagging their samples as a spike-contamination; a dilation of ± 15 samples is applied around each flagged index, producing a binary exclusion mask over the 1000-sample window.

Stage 2 — Quiroga MAD on the clean set. If the unmasked set x_{clean} , that now only contains the signal without spike-contamination, retains at least 200 samples (20% of the window), the Quiroga estimator is applied to it:

$$\sigma_{\text{clean}} = \frac{\text{median}(|x_{\text{clean}} - \text{median}(x_{\text{clean}})|)}{0.6745} \quad (5.4)$$

providing a substantially less biased estimate of the true background noise standard deviation. If fewer than 200 samples survive masking, σ_{perc} is used directly as the final estimate, guaranteeing that the algorithm always produces a usable threshold.

The detection threshold voltage for each channel is then:

$$V_{\text{th}} = \text{median}(x) + \max(K_{\text{Quiroga}} \cdot \sigma, 10 V_{\text{LSB}}) \quad (5.5)$$

where σ is σ_{clean} or σ_{perc} as above, $V_{\text{LSB}} = 3.3/256 \approx 12.9 \text{ mV}$ is one DAC LSB (Section 3.2), and the $10 V_{\text{LSB}}$ floor prevents the threshold from rounding to zero on exceptionally quiet channels. The voltage is then quantized to the 8-bit DAC code transmitted to the FPGA:

$$\text{DAC_code} = \text{clamp}\left(\text{round}\left(\frac{V_{\text{th}} \times 256}{3.3}\right), 0, 255\right) \quad (5.6)$$

The GUI transmits the resulting 60 bytes sequentially; the FPGA programs the DACs channel by channel (Section 4.3), completing the configuration of the 60 Schmitt trigger reference voltages. The GUI then sends the 3-byte start command [0x01, MSB, LSB] where the payload encodes `user_counter_end` (Equation 5.1), and the FPGA transitions to acquisition mode.

5.3 Asynchronous Event Acquisition

Once the start command has been transmitted, the GUI registers an asynchronous serial callback triggered every 600 incoming bytes and starts a 50 ms graphics timer.

Packet decoding. For each packet, the first 4 bytes are assembled MSB-first into the 32-bit raw timestamp, and bytes 5–12 into a 64-bit channel word:

$$\text{ts_raw} = \text{uint32}(\text{bytes } 1\text{--}4) \quad (5.7)$$

$$\text{valori_u64} = \text{uint64}(\text{bytes } 5\text{--}12) \quad (5.8)$$

The state of user-selected channel c (0-indexed) is extracted as `bitget(valori_u64, c+5)`, where channel 0 maps to bit 5 and channel 59 to bit 64 (MATLAB 1-indexed) of `valori_u64`. The 4-bit parity field occupies bits [4 : 1] and is currently not checked by the GUI (Section 4.7.3).

Framing integrity. In steady state, consecutive packet timestamps must differ by the sampling period expressed in microseconds:

$$\Delta_{\text{exp}} = \text{round}\left(\frac{T_s[\text{ns}]}{1000}\right) \quad (5.9)$$

A measured delta $d = \text{ts_raw}[n] - \text{ts_raw}[n - 1]$ (unsigned 32-bit arithmetic) is accepted as valid if:

$$\text{valid}(d) \iff d \neq 0 \wedge d \leq 10 \Delta_{\text{exp}} \wedge (d \bmod \Delta_{\text{exp}} \leq 1 \vee d \bmod \Delta_{\text{exp}} \geq \Delta_{\text{exp}} - 1) \quad (5.10)$$

The $\pm 1 \mu\text{s}$ tolerance absorbs rounding in Δ_{exp} ; the $10 \Delta_{\text{exp}}$ ceiling rejects deltas large enough to signal a framing error rather than a legitimately missed sample.

While the stream is synchronized, the callback validates the delta between each consecutive pair of decoded timestamps against Equation 5.10. The first invalid delta truncates the accepted batch and marks the stream as unsynchronized. If that occurs, the callback shifts the buffer start one byte at a time, and at each candidate offset, it extracts three timestamps spaced 12 bytes apart; the first offset that yields two valid deltas is accepted, and synchronization is restored. Otherwise, if no compliant offset is found, the last 35 bytes are retained and the search is deferred to the next invocation.

Absolute timestamp reconstruction and wraparound. In order to mitigate the fact that the 32-bit `ts_raw` field overflows every $2^{32} \mu s \approx 71.6 \text{ min}$ (Section 4.2), it has been implemented a wraparound correction to produce a monotonically increasing absolute timestamp. Two state variables are maintained across callbacks: `ts_offset`, a cumulative multiple of 2^{32} initialized to zero, and `ts_last`, the final raw timestamp of the previous callback. At the start of each batch, if the first raw timestamp is smaller than `ts_last`, a wrap has occurred between callbacks and `ts_offset` is incremented by 2^{32} . Within the batch, any reversal `ts_raw[k] < ts_raw[k - 1]` is handled analogously, incrementing `ts_offset` before the absolute timestamp of sample k is computed.

5.4 Refractory period filter

The absolute refractory period of cortical neurons, during which the generation of a second action potential is physically unrealistic [8], lasts approximately 2 ms; it is therefore assumed that no further detection events can occur on the same channel within this interval. However, a single action potential may produce multiple threshold crossings due to two distinct mechanisms: the intrinsic afterhyperpolarization of the spike waveform, and residual ringing, consisting of damped transient oscillations in the analog front-end circuitry induced by the rapid spike transient.

Since the majority-vote debouncer operates on a nanosecond timescale and cannot resolve such threshold re-crossings at the millisecond scale, and since no genuine second event can occur within this physiological window, a per-channel refractory filter is applied after packet decoding. Denoting by $M_{p,c}$ the decoded event bit for packet p and channel c , by t_p the absolute timestamp of packet p , and by t_c the timestamp of the most recently accepted event on channel c (initialized to $-\infty$):

$$M_{p,c} \leftarrow \begin{cases} 0 & \text{if } t_p - t_c \leq T_{\text{refr}} \\ 1, \quad t_c \leftarrow t_p & \text{otherwise} \end{cases} \quad (5.11)$$

Setting $T_{\text{refr}} = 0$ disables the filter.

After the refractory filter, the callback writes one text row per valid packet: the absolute timestamp in microseconds followed by the binary event bits for each selected channel. Suppressed events are written as zeros rather than omitted, preserving the fixed-rate timeline so that any post-session analysis can distinguish silence from suppression.

5.5 Real-Time Visualization

The visual interface is governed by a `timer` object that fires every 50 ms and drains the plotting buffers accumulated since the previous tick, rendering up to 500 data points per frame. Each selected channel output is plotted: a dot is placed at the channel baseline for an inactive sample (event bit = 0) and at a raised position for a detected event (event bit = 1), with a vertical stem connecting baseline to dot in the latter case. To stop the acquisition, a red *Stop & Reset* button terminates the session by transmitting the 6-byte stop command to the FPGA (Section 4.7.3), closing the log file, and releasing the serial port.

Part IV

Experimental Protocol and Validation Methods

Chapter 6

Experimental Protocol

This chapter describes the methodology adopted to validate the NeuERA platform. All studies were conducted using simulated spike signals generated by a calibrated data acquisition device rather than a live neuronal culture, so that every detection could be compared against a known ground truth. The experimental protocol comprises four independent studies. Study 1 maps the detection performance over the joint space of threshold multiplier, signal-to-noise ratio, and firing rate. Study 2 tests how far performance degrades when the spike amplitude drifts away from the calibration amplitude within a single session, and Study 3 repeats the analysis under biologically realistic Poisson spike timing. Study 4 characterizes the onboard temporal latency of the acquisition pipeline.

6.1 Hardware Setup and Signal Injection

The validation platform consists of three components operating concurrently from the same host PC: the NeuERA acquisition system (Chapters 3–5), a National Instruments USB-6259 data acquisition device (DAQ), and the MATLAB environment managing both.

During validation the microelectrode array, neuronal culture, and AFE boards (Section 3.2) are removed from the signal path. Instead of AFEs, passive bridges are used to provide mechanical and electrical continuity between the MEA interface connector and the motherboard. Through these bridges, a synthetic signal generated by the DAQ analog output, connected via a dedicated PCB to the Motherboard, at 25 kS/s is driven directly into the INAs input, completely bypassing any initial amplification or filtering. From that point, the signal follows the standard acquisition path described in Section 3.2.

Rather than injecting the signal at its native microvolt scale, the DAQ delivers it at the millivolt-level amplitude that the signal would already have after full AFE pre-amplification.

Moreover, two channels — CH 9 and CH 29 — are excluded from all analyses since CH 9 is used as ground and CH 29 showed no detectable event activity across all acquisitions due to a hardware malfunction. Therefore, the effective channel count is 58 out of 60.

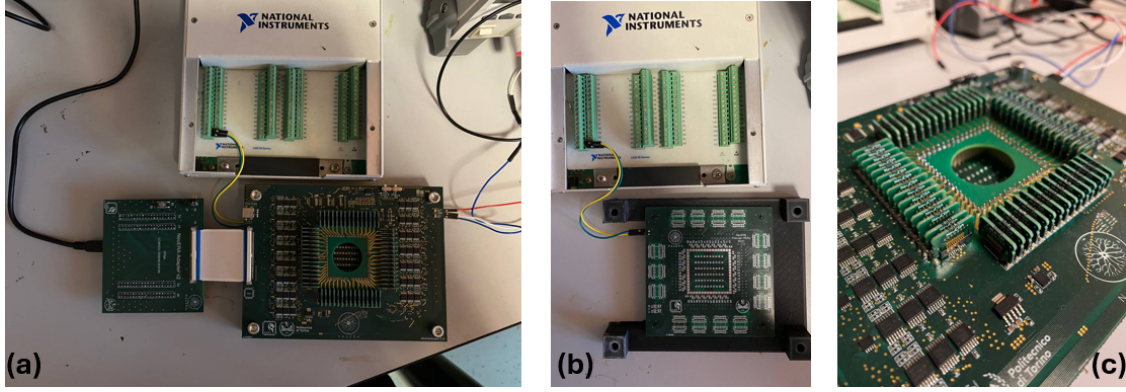


Figure 6.1. **Experimental validation setup.** (a) Complete measurement setup, showing the motherboard, the NI USB-6259 DAQ, and the FPGA adapter connected to the motherboard via an FFC connector. (b) The DAQ and the dedicated PCB used to inject the synthetic signal into all channels. (c) Detail of the passive bridge that replaces the AFE during experimental validation.

6.2 Synthetic Spike Model and Acquisition Structure

6.2.1 Synthetic Spike Model

All four studies use the same waveform, whose aim is to reproduce the biphasic shape of an extracellular action potential recorded after AFE amplification, modelled as an overlap of two Gaussian pulses of opposite polarity:

$$s(t) = \exp\left(-\frac{(t - T/4)^2}{2\sigma_1^2}\right) - 0.5 \cdot \exp\left(-\frac{(t - 3T/4)^2}{2\sigma_2^2}\right), \quad t \in [0, T] \quad (6.1)$$

with $T = 2$ ms, $\sigma_1 = 0.3$ ms, and $\sigma_2 = 0.4$ ms. The positive lobe peaks at $t = T/4 = 0.5$ ms with unit amplitude; the negative lobe reaches -0.5 at $t = 3T/4 = 1.5$ ms. The template is sampled at the DAQ output rate of 25 kS/s, yielding 50 samples per spike.

Additive white Gaussian noise is superimposed on the spike train before transmission to the DAQ:

$$x(t) = A_{\text{spike}} \sum_k s(t - t_k) + \sigma_{\text{noise}} \cdot w(t) \quad (6.2)$$

where $\{t_k\}$ are the spike onset times, $w(t) \sim \mathcal{N}(0,1)$ is a white Gaussian noise process, and A_{spike} is the peak amplitude in millivolts. The signal-to-noise ratio is defined as the linear amplitude dimensionless ratio:

$$\text{SNR} = \frac{A_{\text{spike}}}{\sigma_{\text{noise}}} \quad (6.3)$$

Figure 6.2 shows an example of the composite signal $x(t)$ at the reference parameters used throughout the studies.

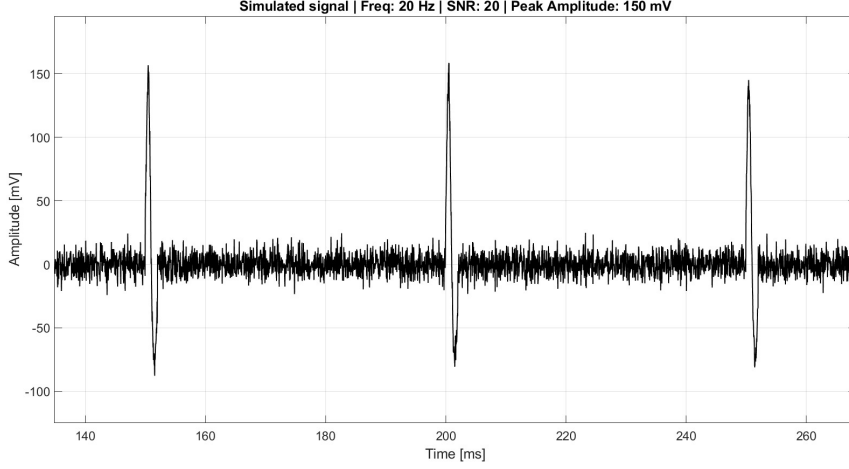


Figure 6.2. Example of the synthetic composite signal $x(t)$ injected by the DAQ (Equation 6.2). Parameters: $A_{\text{spike}} = 150 \text{ mV}$, $\text{SNR} = 20$ (linear), $\text{FiringRate} = 10 \text{ Hz}$ (fixed inter-spike interval).

6.2.2 Alignment preamble and Acquisition Structure

In order to overcome the issue that both FPGA and DAQ operate on independent crystal oscillators with no hardware synchronization line, every acquisition begins with a preamble based on the 13-bit Barker sequence $\mathbf{b} = [1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1]$ [43], where 1 bit represents a spike injection, whilst 0 bit contains only noise. Each bit occupies a $T_{\text{bit}} = 15 \text{ ms}$ slot, spanning 195 ms.

Then, every acquisition consists of three consecutive phases transmitted as a single continuous waveform, which are summarized and described as follows:

Table 6.1. Temporal structure of a single acquisition.

Phase	Duration	Description
Barker-13 preamble	195 ms	9 spikes at $\text{SNR} = 20$ for temporal lock
Guard band	20 ms	Silence separating preamble from target
Target signal	60 s	Acquisition window for performance evaluation
Analysis window	55 s	Target with last 5 s excluded

6.3 Experimental Design

Study 1 analyzes the detection performances across the full joint space of threshold multiplier K_{Quiroga} , signal-to-noise ratio (SNR), and firing rate f_r , whose reciprocal defines the inter-spike interval, $\text{ISI} = 1/f_r$. Results obtained in this study determine the values held fixed in Studies 2 and 3, which then extend the analysis to two dimensions not covered by Study 1: robustness to amplitude mismatch between calibration and acquisition, and

behaviour under biologically realistic Poisson spike timing, respectively. Finally, Study 4 characterizes the onboard temporal latency of the acquisition pipeline, distinguishing the FPGA-internal contribution from the end-to-end system latency, which additionally includes the analog signal path.

6.3.1 Study 1 — $K \times \text{SNR} \times f_r$ Sweep

Study 1 characterizes the detection performance of the platform as a function of three parameters that directly govern the threshold-based event detection:

- **The Quiroga multiplier (K_{Quiroga})**, which sets the threshold level relative to the estimated background noise.
- **The signal-to-noise ratio (SNR)**, which represents the stimulus intensity relative to that noise.
- **The mean spike firing rate (f_r)**, which determines the temporal density of events.

Table 6.2 reports the variable parameter grid, whilst all other variables not mentioned are held constant. The refractory filter (Section 5.4) is disabled during acquisition ($T_{\text{refr}} = 0$) and applied with $T_{\text{refr}} = 2$ ms in the offline analysis.

Table 6.2. Parameter grid for Study 1.

Parameter	Values	Notes
K_{Quiroga}	{3, 4, 5}	threshold multiplier
SNR (linear)	{3, 4, 6, 8, 10, 20}	$A_{\text{spike}}/\sigma_{\text{noise}}$
f_r	{5, 10, 20, 50, 100, 200, 300} Hz	mean firing rate
A_{spike}	150 mV	fixed
Sessions	5	independent repetitions

The full study comprises $3 \times 6 \times 7 = 126$ parameter combinations per session, for a total of $126 \times 5 = 630$ acquisitions.

Each acquisition is preceded by an independent calibration, ensuring that the DAC thresholds are always computed with the current signal in analysis. The statistical analysis of Study 1 is described in Section 7.2.

6.3.2 Study 2 — Amplitude Sweep with Fixed Calibration

Study 2 analyzes the robustness of the detection when the spike amplitude at acquisition time differs from the amplitude present during calibration, since in a real recording session the signal amplitude can vary due to electrode drift, neuronal adaptation, or changes in tissue coupling. The described scenarios are simulated by calibrating once at a reference amplitude and then testing at a range of amplitudes, including conditions both below and above the calibration point.

K_{Quiroga} , SNR and f_r are fixed at the values identified from the analysis of the results obtained in Study 1. Unlike Study 1, calibration is performed once per session at $A_{\text{ref}} =$

150 mV and the resulting DAC thresholds have been unchanged for all five amplitude conditions. As well as in Study 1, the refractory filter (Section 5.4) is disabled during acquisition ($T_{\text{refr}} = 0$) and applied with $T_{\text{refr}} = 2$ ms in the offline analysis.

Table 6.3. Parameter grid for Study 2.

Parameter	Values	Notes
A_{spike}	{50, 75, 100, 125, 150} mV	swept parameter
K_{Quiroga}	4	fixed (from Study 1)
Calibration at	$SNR = 10$	fixed (from Study 1)
f_r	50 Hz	fixed (from Study 1)
Calibration at	$A_{\text{ref}} = 150$ mV	once per session
Sessions	5	independent repetitions

The full design comprises $5 \times 5 = 25$ acquisitions and the estimated noise standard deviation at calibration time is $\sigma_{\text{cal}} = A_{\text{ref}}/SNR = 15$ mV, producing thresholds approximately around $K_{\text{Quiroga}} \cdot \sigma_{\text{cal}} = 60$ mV above the channel baseline. It is important to specify that at test amplitudes below A_{ref} , the SNR decreases proportionally, but the threshold remains fixed, thus placing the evaluation in an extremely unfavorable measurement condition at low amplitudes.

6.3.3 Study 3 — Poisson Firing Rate

Studies 1 and 2 use a predetermined, specific timing between consecutive spikes, whilst Study 3 employs a random, stochastic timing of spikes according to a Poisson process, thereby creating a more realistic model of the irregular, variable firing patterns of cortical neurons. The spike train is generated by drawing inter-spike intervals from an exponential distribution:

$$\Delta t_k \sim \text{Exp}\left(\frac{1}{f_r}\right), \quad t_k = \sum_{i=1}^k \Delta t_i \quad (6.4)$$

where f_r is the mean firing rate in Hz.

Thus, the aim of this study is to establish whether the same level of detection accuracy achieved under deterministic timing also holds under realistic timing variability, and to determine how burst clustering of spikes interacts with the proposed acquisition pipeline. The parameters associated with the spikes are as defined for Study 1, with the only exception of the mean spike rate and the timing model.

The design comprises $7 \times 5 = 35$ acquisitions. As in Study 1, an independent calibration is performed before every acquisition following Section 5.1, with $K_{\text{Quiroga}} = 4$ and $A = 150$ mV held fixed throughout. Each acquisition follows the standard structure defined in Section 6.2. Figure 6.3 shows an example of the resulting spike train at the reference parameters.

Table 6.4. Parameter grid for Study 3.

Parameter	Values	Notes
Mean f_r	{5, 10, 20, 50, 100, 200, 300} Hz	swept parameter
Timing model	Poisson (exponential ISI)	vs. fixed ISI in Study 1
K_{Quiroga}	4	fixed (from Study 1)
SNR (linear)	10	fixed (from Study 1)
A_{spike}	150 mV	fixed
Sessions	5	independent repetitions

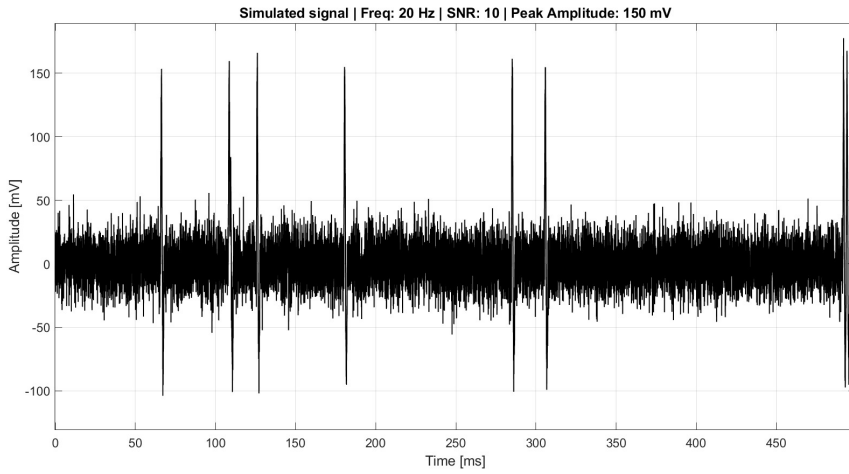


Figure 6.3. Example of the synthetic signal $x(t)$ with Poisson-distributed inter-spike intervals (Equation 6.4). Parameters: $A_{\text{spike}} = 150$ mV, SNR = 10 (linear), mean firing rate $f_r = 20$ Hz. The irregular spacing between spikes, characteristic of the exponential ISI distribution, is clearly visible compared with the regular timing of Studies 1 and 2.

6.3.4 Study 4 — Onboard Latency Characterization

Study 4 measures two latency contributions along the acquisition pipeline: the delay internal to the FPGA and the delay of the full chain from the DAQ output to the FPGA. The former is contained within the latter, so the two are measured against the same GPIO reference edge. To expose that edge, a minor firmware modification was introduced in `main_module`, routing the `sticky_mask` output (Section 4.5) of channel 33 — selected empirically — to a dedicated GPIO probe signal. The first evaluation is the **FPGA-internal latency** (τ_{FPGA}), which is defined as the time elapsed between the comparator signal entering the FPGA and the assertion of this GPIO edge. The second is the **end-to-end system latency** (τ_{system}), defined as the total propagation delay from spike generation at the DAQ output to the assertion of the same GPIO edge, thereby encompassing both the analog signal path and the FPGA-internal contribution jointly.

FPGA-Internal Latency

The measurement was performed under fixed conditions, reported in Table 6.5, using the oscilloscope’s Statistics mode at a temporal resolution of 50 ns/div, sufficient to resolve latencies on the order of tens of nanoseconds.



Figure 6.4. **Oscilloscope measurement of the FPGA-internal latency (τ_{FPGA})**. CH1 (yellow) probes the comparator signal entering the FPGA input pin; CH4 (blue) probes the corresponding GPIO edge on the `sticky_mask` output. The bottom-left panel reports the Delay 1→4 statistics accumulated over the acquisition (Cur, Avg, Max, Dev, Cnt), from which the per-session mean and standard deviation reported in Table 8.9 are extracted.

Table 6.5. Measurement parameters for the FPGA-internal latency (τ_{FPGA}).

Parameter	Values	Notes
A_{spike}	150 mV	fixed
SNR (linear)	10	fixed
f_r	50 Hz	fixed
Probed channel	33	empirically selected
Oscilloscope resolution	50 ns/div	Statistics mode
Repetitions	15	independent measurements

The design comprises 15 independent measurements, each one preceded by an independent calibration following with the parameters listed in Table 6.5 held fixed throughout.

End-to-End System Latency

The end-to-end system latency τ_{system} was characterized combining two values of the threshold multiplier K_{Quiroga} with three levels of SNR, as reported in Table 6.6. The oscilloscope

sampled at 100 kHz, corresponding to a temporal resolution of $10\ \mu\text{s}$, adequate to resolve latencies on the order of hundreds of microseconds.

Table 6.6. Parameter grid for the end-to-end system latency (τ_{system}).

Parameter	Values	Notes
K_{Quiroga}	{3, 4}	swept parameter
SNR (linear)	{8, 10, 20}	swept parameter
f_r	50 Hz	fixed
Session duration	$\approx 10\text{ s}$	per session
Sessions	10	independent repetitions per condition
Oscilloscope sampling rate	100 kHz	$10\ \mu\text{s}$ resolution

For each of the $2 \times 3 = 6$ combinations, ten independent sessions were recorded, each approximately 10 s long at a spike rate of 50 Hz, yielding approximately 500 events per session and 5000 events per condition. Each session is preceded by an independent calibration, ensuring that the DAC thresholds are always computed with the current signal in analysis.

Each recorded CSV file was processed offline through a four-stage MATLAB pipeline. First, the raw DAQ signal was preprocessed by removing the DC offset and applying a 4th-order zero-phase Butterworth low-pass filter at 3 kHz. Peaks were then detected on the filtered signal using `findpeaks`, with a minimum amplitude threshold of 50 mV and a minimum inter-peak distance of 2 ms, the latter implicitly enforcing a refractory constraint that prevents double-counting of a single spike. GPIO edges were independently identified by thresholding the CH2 signal at 2.5 V to detect LOW-to-HIGH transitions. Finally, each DAQ peak was matched to the earliest available GPIO edge within the subsequent 5 ms window. Statistical testing for Study 4 is described in Section 7.2.

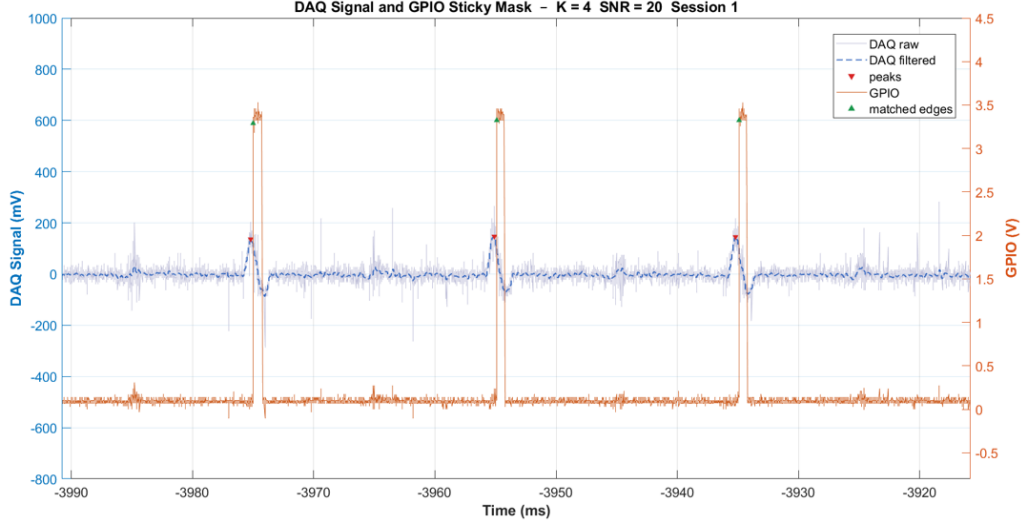


Figure 6.5. **Example of the DAQ-to-GPIO peak matching procedure** ($K_{\text{Quiroga}} = 4$, $\text{SNR} = 20$, session 1). Left axis: raw (light blue) and filtered (dashed blue) DAQ signal, with detected peaks marked by red triangles. Right axis: GPIO `sticky_mask` trace (orange). Green triangles mark the matched (DAQ peak, GPIO edge) pairs used to compute τ_{system} for this session.

6.4 Temporal Alignment and Clock Skew Correction

Studies 1–3 require a temporal alignment procedure before metric computation, since both the FPGA and the DAQ operate on independent crystal oscillators with no hardware synchronisation line. In order to align all channels using the same offset, a signal representative of the entire acquisition session was generated, named as Multi-Unit Activity (MUA), which evaluates whether, within a given time window, simultaneous activity has occurred across a known number of channels belonging to the population under analysis. This approach is justified by the fact that all channels receive the same signal at the same time instant and, therefore, all events should theoretically be detected with the same timestamp. Applying this method, a packet timestamp is considered to belong to the MUA signal if the number of channels that detected an event simultaneously is equal to or greater than N_{min} . This approach is adopted, with different thresholds, both in step 1, where the offset is searched for by aligning the Barker signal described in Section 6.2, and in step 4, where the correction is performed via linear regression, using a higher threshold on the number of channels involved and on the signal under evaluation. Alignment can be described in the following four sequential steps:

- **Step 1 — Barker direct lock:** The MUA stream is searched for the constant temporal offset δ such that, when the Barker ground-truth spike times $\{t_k^B\}$ are shifted by δ , the maximum number of them fall within ± 1 ms of an MUA event. A candidate offset is accepted if at least 6 of the 9 Barker synchronisation spikes are matched ($\text{MIN_MATCH} = 6/9$).

- **Step 2 — Validation on the target signal:** The candidate offset from Step 1 is validated by applying it to the first 60 MUA ($N_{\min} = 15$) events in the target window and checking that at least 25% fall within ± 1 ms of a ground-truth spike time. In case that the match rate falls below the threshold, the candidate offset is rejected and the pipeline proceeds to Step 3.
- **Steps 3–4 — Distributed fallback and Barker + prior:** If validation fails, a distributed offset is estimated by exhaustive cross-correlation between the strict MUA stream and the ground-truth spike train over the hardware-constrained search range $[-400, -50]$ ms. The newly obtained offset is then reused to verify whether, with the correction applied, Step 1 — i.e., the lock via the Barker signal — still succeeds. If the test is positive, the corrected offset is considered validated; otherwise, the distributed offset computed via cross-correlation is used. This step represents a fallback that is nonetheless triggered very rarely, owing to the high reliability of the primary locking method; indeed, across the acquisitions analyzed, the activation rate was found to be close to 0%.
- **Clock skew correction:** After the offset is established, a linear regression is performed between the raw FPGA event times (strict MUA, target window) and their nearest matched ground-truth counterparts:

$$t_{\text{GT}} = \alpha \cdot t_{\text{FPGA}} + \beta \quad (6.5)$$

At least 50 matched pairs are required; the tolerance for pairing is $\min(2 \text{ ms}, 0.15 T_{\text{ISI}})$. The regression is accepted only if the residual between the Barker-derived offset and the polyfit intercept satisfies $|\delta_{\text{Barker}} - \beta| \leq 3 \text{ ms}$, corresponding to the maximum drift expected from a 50 ppm crystal over a 55 s window ($50 \times 10^{-6} \times 55 \text{ s} \approx 2.75 \text{ ms}$). All per-channel FPGA timestamps are then corrected as $t_{\text{corr}} = \alpha \cdot t_{\text{FPGA}} + \beta$ before metric computation.

Figure 6.6 illustrates the complete alignment result on a representative single-channel acquisition.

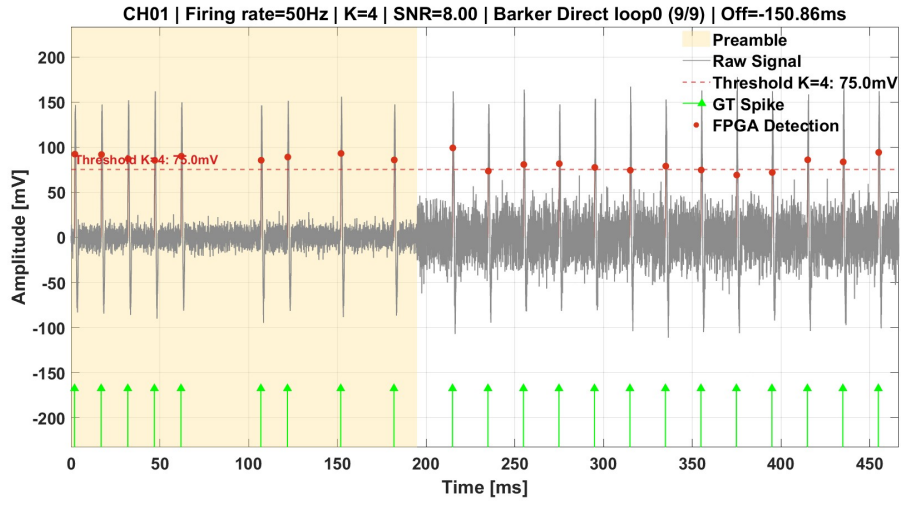


Figure 6.6. Temporal alignment result for a representative acquisition (CH01, $f_r = 50$ Hz, $K_{\text{Quiroga}} = 4$, $\text{SNR} = 8$). The orange-shaded region marks the Barker-13 preamble (0–195 ms); the white region is the target signal window. The red dashed line is the calibrated detection threshold (75 mV). Green arrows indicate ground-truth spike onset times; red circles mark the corresponding FPGA detections after offset correction.

Chapter 7

Validation Methods

7.1 Evaluation Metrics

The metrics defined in this section are organised into two groups. The first covers Studies 1–3 and is based on the comparison between FPGA detections and software-aligned ground-truth spike times. The second covers Study 4 and characterizes the onboard propagation latency measured directly via oscilloscope.

The FPGA detects a spike when the conditioned signal crosses the DAC threshold on its rising edge, which usually occurs before the waveform reaches its positive peak. However, to evaluate the following metrics, the ground-truth onset times are shifted by $+T/4$ before matching:

$$t_k^{\text{peak}} = t_k + \frac{T}{4} = t_k + 0.5 \text{ ms} \quad (7.1)$$

since the peak is the most informative point of the neural activation and less susceptible to noise than the threshold crossing [44], and because it lets the ± 1 ms tolerance mask be applied symmetrically. If the mask were centred on the rising edge instead, its -1 ms bound would fall before the edge, in a window where no spike can yet be present, while its $+1$ ms bound would not reach the end of the spike. Centring the mask on the peak covers the full duration of the neural activation. This convention applies to Studies 1–3 for ground-truth matching, and to Study 4 where τ_{system} is defined with respect to the DAQ spike peak by construction [45], consistent with the convention adopted by comparable FPGA-based closed-loop acquisition systems [46, 47].

Detection and timing metrics (Studies 1–3). Performance is evaluated independently for each channel and each acquisition. For every channel c , the corrected FPGA detections are matched against the ground-truth peak times using a greedy minimum-distance algorithm: all candidate pairs (j, k) with $|t_j^{\text{corr}} - t_k^{\text{peak}}| \leq \tau$ are ranked by absolute error and accepted in ascending order, with each detection and each ground-truth spike used at most once. The matching tolerance is $\tau = 1$ ms, consistent with the FPGA sampling period T_s .

It is therefore possible to calculate true positives (TP), false positives (FP), and false

negatives (FN) which lead to the following detection metrics:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}, \quad F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (7.2)$$

- **Precision** measures the fraction of FPGA detections that correspond to a real spike — high Precision indicates few false alarms.
- **Recall** measures the fraction of real spikes that were detected — high Recall indicates few missed events.
- **F-Score** F_1 is the harmonic mean of the two and synthesizes the detection-missed trade-off into a single scalar; it is preferred over the arithmetic mean because it penalizes imbalance between Precision and Recall. Channels with no detections contribute $F_1 = 0$ to the session average rather than being silently excluded.

For channels with at least three matched TP pairs, two timing metrics are computed on the signed errors $e_j = t_j^{\text{corr}} - t_j^{\text{peak}}$:

- **Latency** is the mean of $\{e_j\}$ and quantifies the systematic advance or delay of the FPGA detection relative to the spike peak.
- **Jitter** is the standard deviation of latency and captures the residual temporal uncertainty of the detection system after alignment.

Since the offset correction alters the latency values, latency is not evaluated as a performance metric; it is used only to compute jitter, which, being the standard deviation of latency, is unaffected by the offset correction and therefore remains an accurate and evaluable measure. The onboard propagation latency is instead characterized directly in Study 4, where the absolute delay is measured via oscilloscope without any software-based temporal alignment, making it immune to this limitation.

Onboard latency metrics (Study 4). For the FPGA-internal latency, each of the $N = 15$ oscilloscope repetitions yields a single measurement $\tau_{\text{FPGA}}^{(i)}$; the reported value is the sample mean $\bar{\tau}_{\text{FPGA}}$ and standard deviation $\sigma_{\tau_{\text{FPGA}}}$ across repetitions (with the conventional $N - 1$ denominator).

For the end-to-end system latency, each matched pair obtained through the matching procedure (Section 6.3.4) yields a single-event latency $\tau_{\text{system}}^{(j)} = t_{\text{GPIO}}^{(j)} - t_{\text{peak}}^{(j)}$. For each of the six $\text{SNR} \times K_{\text{Quiroga}}$ conditions, $\bar{\tau}_{\text{system}}$ and $\sigma_{\tau_{\text{system}}}$ are computed analogously, over the full population of matched events within that condition.

7.2 Statistical Analysis

The statistical analysis described in this section applies to Study 1 and Study 4, whilst Studies 2 and 3 are evaluated using descriptive statistics only.

7.2.1 Study 1

Unit of analysis. The unit of observation for all inferential tests is the *session-level mean* $\bar{F}_1^{(s)}$, defined as the arithmetic mean of per-channel F_1 values over the 58 valid channels within a single acquisition, since all channels share the same applied synthetic signal and are therefore characterised by the same signal-to-noise ratio (SNR), the same amplitude, the same session-level environmental conditions, and mutually similar calibration thresholds, save for differences arising from the intrinsic properties of the electrodes, which may result in greater or lesser sensitivity to noise. This gives rise to an intra-session correlation that prevents each channel from being treated as an independent statistical observation. Therefore, values are averaged at the session level and the per-session F_1 mean is taken as the statistical unit, yielding a total of five observations per experimental condition, placing the study in a conservative position.

Repeatability. Before testing the effect of K_{Quiroga} , in order to apply the mean, the inter-session repeatability is quantified by the coefficient of variation:

$$\text{CV} = \frac{\text{std}(\bar{F}_1^{(1)}, \dots, \bar{F}_1^{(5)})}{\text{mean}(\bar{F}_1^{(1)}, \dots, \bar{F}_1^{(5)})} \times 100\% \quad (7.3)$$

A condition is considered stable if $\text{CV} < 10\%$, i.e. the session-to-session variability is less than one tenth of the mean performance.

Effect of K_{Quiroga} — Friedman test and FDR correction. For each of the $6 \times 7 = 42$ (SNR, f_r) conditions, the five session means are arranged in a 5×3 block matrix (sessions $\times$ K values) and submitted to the Friedman test [48], a non-parametric rank test for repeated measures. Within each session, the three K values are ranked, and the test compares the resulting rank sums against a χ^2 distribution to decide whether at least one K value differs systematically from the others. A parametric alternative was avoided because the F_1 distribution is bounded in $[0,1]$ and, with only five observations per condition, normality cannot be assumed.

Because 42 simultaneous tests are performed, the raw p -values were corrected to control the false discovery rate (FDR), which quantifies the expected proportion of false positives among all conditions declared significant. Setting $\text{FDR} = 0.05$, at most 5% of the conditions declared significant are expected to constitute a false discovery. To implement FDR control, the Benjamini–Hochberg procedure [49] was adopted, which sorts the 42 obtained p -values in ascending order and computes, for each k -th position, an adaptive threshold equal to

$$\frac{k\alpha}{m}, \quad (7.4)$$

where $\alpha = 0.05$ and $m = 42$ is the total number of comparisons performed. Only conditions for which the observed p -value is less than or equal to the corresponding adaptive threshold are then declared significant. As is evident from the formulation, the threshold grows linearly with the position of the ordered p -value; consequently, conditions supported by very strong statistical evidence are declared significant even under very stringent absolute thresholds, whereas borderline evidence tends not to clear the selection criterion.

Effect size — Kendall’s W . The Friedman p -value gives no information about how consistently sessions agree in ranking the three K values. For this reason, Kendall’s coefficient of concordance, W , is used: it analyses each session separately, ranking the mean F_1 of the three K values to obtain a session-specific ranking. Repeated across all 5 sessions, this yields 5 independent rankings, and W measures the extent to which these 5 rankings coincide. It ranges between 0 and 1: $W = 0$ means that sessions produce completely uncoordinated orderings, while $W = 1$ means that all 5 sessions rank K identically. The threshold $W \geq 0.5$ is conventionally classified as a large effect according to the Cohen scale adapted to concordance tests.

For each condition where the Friedman test is significant after FDR correction, the concordance coefficient W is computed as:

$$W = \frac{\chi_{\text{Friedman}}^2}{n(k-1)} \quad (7.5)$$

where $n = 5$ is the number of sessions and $k = 3$ is the number of K values. W ranges from 0 (no agreement) to 1 (perfect agreement in ranking); following the convention adapted from Cohen [50], values $W \geq 0.5$ are classified as a large effect.

The relationship with the Friedman test is algebraic, not logical: the two indices are computed on the same 5×3 matrix but capture different aspects. The Friedman test asks whether the rank sums are dispersed enough to rule out chance, while W quantifies how dispersed they are on a 0–1 scale. This distinction matters because, with only 5 sessions, the Friedman test has limited statistical power and may miss a real effect, whereas W measures concordance independently of that power. The K_{Quiroga} value achieving the highest session-mean F_1 is recorded as the optimal value for that condition.

7.2.2 Study 4

Unit of analysis. Within each repetition, the three SNR conditions are recorded in rapid succession and therefore share the same ambient conditions. They constitute repeated measures rather than independent observations, with the 10 repetitions acting as statistical blocks. Consequently, the unit of observation for all inferential tests is the *session-level median* $\tilde{\tau}_{\text{system}}^{(s)}$, computed separately for each of the 10 repetitions and yielding ten paired observations per K_{Quiroga} value. The median, rather than the mean, is adopted here since the propagation-delay distribution is asymmetric.

Effect of SNR — Friedman test and Kendall’s W . For each value of K_{Quiroga} , the 10 session medians are arranged in a 10×3 block matrix (repetitions $\times$ SNR levels) and submitted to the Friedman test [48], following the same rank-based procedure described in Section 7.2.1. Kendall’s coefficient of concordance W (Equation 7.5, with $n = 10$ and $k = 3$) is computed alongside each test to quantify the consistency of the SNR ranking across repetitions.

Effect of K_{Quiroga} — Wilcoxon signed-rank test. Because the design includes only two levels of K_{Quiroga} , a Friedman test would rely on a χ^2 approximation that is anti-conservative for such a small number of treatments. The paired comparison between

$K_{\text{Quiroga}} = 3$ and $K_{\text{Quiroga}} = 4$ is therefore performed with the Wilcoxon signed-rank test [51]: for each of the ten repetitions, the signed difference between the two session medians is computed and ranked in absolute value, and the ranks of the positive and negative differences are summed separately to obtain the statistics T^+ and T^- . Given $n = 10$ paired observations, the exact distribution of T^+ is used rather than the Gaussian approximation, which is more reliable for small samples. The associated effect size, $r = (T^+ - T^-)/(T^+ + T^-)$, ranges from -1 to 1 and reaches unit magnitude when every paired difference favours the same condition.

Post-hoc pairwise comparisons and multiple-comparison correction. Because the Friedman test is an overall test, a significant result indicates that at least one SNR condition differs from the others without identifying which pair. Post-hoc pairwise comparisons were therefore performed for the three possible SNR pairs (20 vs 10, 20 vs 8, 10 vs 8), using the same exact Wilcoxon signed-rank test described above, separately for each value of K_{Quiroga} . Since six comparisons are performed overall (three SNR pairs $\times$ two K_{Quiroga} values), the resulting p -values are corrected using the Benjamini–Hochberg procedure [49] described in Section 7.2.1, with $m = 6$ and $\text{FDR} = 0.05$.

Part V

Results, Discussion and
Conclusions

Chapter 8

Results

The following chapter describes the results obtained from the four experimental protocols presented in the previous chapter (Section 6.3). For each experimental condition of Studies 1–3, the parameter means, computed from the mean across the 58 valid channels, together with their standard deviations are reported; Study 4 follows an analogous convention, reporting mean and standard deviation across the underlying set of measurements for each condition. As already mentioned, Study 1 and Study 4 are further examined through the statistical analysis presented in Section 7.2, whereas Studies 2 and 3, which also draw on the results of the first, are described exclusively by means of descriptive metrics

8.1 Study 1 — $K \times \text{SNR} \times f_r$ Sweep

Study 1 evaluates the impact of the threshold multiplier (K_{Quiroga}), the signal-to-noise ratio (SNR), and the firing rate on the absolute detection performance of the NeuERA platform, with the aim of identifying its complete operating range and determining both the optimal and the most critical conditions. The obtained results further allow identification of the parameter values held constant in Studies 2 and 3 (Section 6.3.1).

The following paragraphs report the F_1 -score trend across the SNR– f_r – K experimental space, the inter-session repeatability of these trends, and the statistical analysis of the effect of K_{Quiroga} , concluding with the identification of the optimal K value and the resulting operating range of the device. Finally, an evaluation of the temporal jitter is presented, analysing how variation of the threshold influences this parameter.

8.1.1 Repeatability

Repeatability across the five sessions was quantified using the coefficient of variation (CV), defined in Equation 7.3, and could be computed for 69 of the 126 experimental conditions, whereas for the remaining ones the estimate could not be performed because the F_1 score was zero in every session. These conditions occur systematically at SNR = 3 and SNR = 4, regardless of the K_{Quiroga} value and the firing rate, and more sporadically at the highest firing rate for several (K_{Quiroga} , SNR) combinations. A further case is observed at SNR = 6 for the condition corresponding to $K_{\text{Quiroga}} = 5$.

Among the 69 conditions for which the CV could be computed, its distribution is strongly right-skewed: the median is 1.6%, while the mean is $12.7\% \pm 27.7\%$. Overall, inter-session variability remains below the 10% stability threshold for 55 of the 69 conditions (79.7%).

Figure 8.1 shows the trend of the CV across the experimental conditions, and also reports a white dashed contour delimiting the regions where the CV exceeds 10%. These regions comprise two isolated conditions for $K_{\text{Quiroga}} = 3$, the entire row corresponding to $\text{SNR} = 6$ for $K_{\text{Quiroga}} = 4$, and a localised region around $\text{SNR} = 8$ for $K_{\text{Quiroga}} = 5$, where the highest observed value (154.2%) occurs at $f_r = 200 \text{ Hz}$. The complete list of the 14 conditions exceeding the 10% threshold, together with the corresponding per-session F_1 values, is reported in Table 8.1.

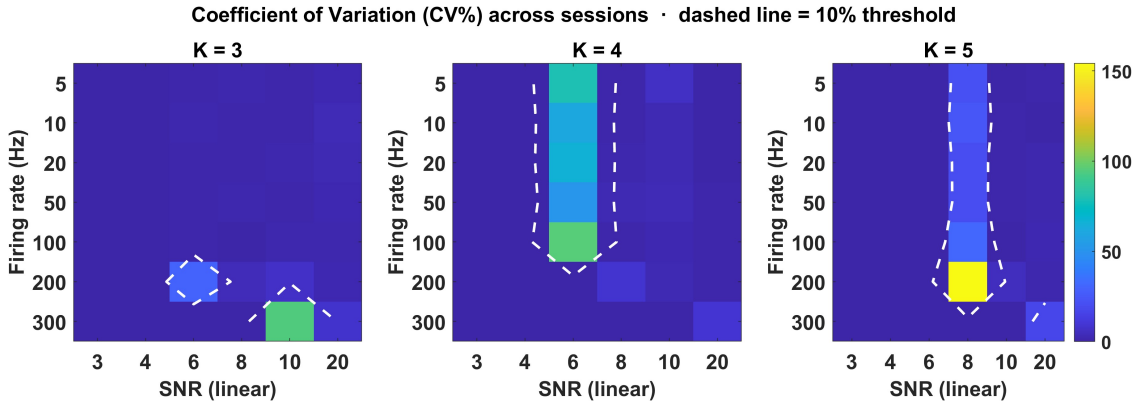


Figure 8.1. Coefficient of variation (CV) of the session-level F_1 -score across the SNR- f_r experimental space, for each K_{Quiroga} value. The white dashed contour marks the 10% stability threshold.

Table 8.1. Per-session F_1 values for conditions exceeding the 10% CV threshold (Study 1 repeatability).

K_{Quiroga}	SNR	f_r (Hz)	CV (%)	$F_1^{(1)}$	$F_1^{(2)}$	$F_1^{(3)}$	$F_1^{(4)}$	$F_1^{(5)}$
3	6	200	29.6	0.186	0.162	0.084	0.111	0.145
3	10	300	94.2	0.100	0.000	0.000	0.086	0.067
4	6	5	79.1	0.253	0.067	0.144	0.049	0.045
4	6	10	61.1	0.201	0.098	0.166	0.047	0.050
4	6	20	66.7	0.260	0.072	0.160	0.067	0.073
4	6	50	51.4	0.153	0.053	0.128	0.069	0.051
4	6	100	95.7	0.095	0.027	0.028	0.020	0.008
5	8	5	21.7	0.736	0.592	0.555	0.436	0.457
5	8	10	24.6	0.797	0.535	0.671	0.490	0.445
5	8	20	20.4	0.758	0.581	0.599	0.489	0.457
5	8	50	20.1	0.782	0.550	0.537	0.476	0.561
5	8	100	31.2	0.609	0.388	0.378	0.359	0.272
5	8	200	154.2	0.003	0.010	0.000	0.001	0.000
5	20	300	17.2	0.947	0.645	0.827	0.673	0.678

8.1.2 Detection Performance

Figures 8.2 and 8.3 describe the trend of the F_1 parameter as a function of f_r and SNR, for the three analysed K_{Quiroga} conditions, whose corresponding mean $\pm$ standard deviation are listed in Tables 8.2–8.4.

For all three conditions, F_1 is zero at SNR = 3 and SNR = 4, and rises sharply towards values close to 1 as the SNR increases, though at different SNR levels for each condition. Indeed, for $K_{\text{Quiroga}} = 3$, this transition occurs between SNR = 4 and SNR = 6, reaching $F_1 \geq 0.92$ already at SNR = 6 for most of the analysed firing rates, unlike $K_{\text{Quiroga}} = 4$, for which F_1 remains below 0.13 at SNR = 6 and exceeds 0.93 from SNR = 8 onward, again for most of the higher firing rates. For $K_{\text{Quiroga}} = 5$, the pattern is the same but shifted towards higher SNR values: F_1 is zero at SNR = 6, reaches an intermediate range of 0.40–0.59 at SNR = 8 depending on the f_r , and exceeds 0.89 from SNR = 10 onward, with the exception of the highest firing rate.

Under operating conditions with high F_1 , a variation in performance is evident at higher f_r levels, both at 200 Hz and at 300 Hz, more pronounced in the latter case: at $f_r = 300$ Hz, F_1 drops to zero for every SNR value up to 10, across all three K_{Quiroga} values, recovering only at SNR = 20, where it reaches 0.947, 0.946, and 0.754 for $K_{\text{Quiroga}} = 3, 4$, and 5 respectively. The decline at $f_r = 200$ Hz is instead less marked but still present, with F_1 remaining above 0.93 for SNR ≥ 10 at $K_{\text{Quiroga}} = 3$ and 4, and above 0.89 at $K_{\text{Quiroga}} = 5$.

As shown in Tables 8.2–8.4, under normal operating conditions the Recall and Precision parameters are balanced, unlike conditions in which F_1 decreases due to low SNR, where precision tends to remain high at the expense of recall, thereby minimising the number of false positives but increasing that of false negatives. In conditions of decline at high firing rate, both precision and recall collapse, as for example at $K_{\text{Quiroga}} = 5$, SNR = 8, $f_r = 200$ Hz, where $F_1 = 0.003$, precision = 0.131, and recall = 0.001.

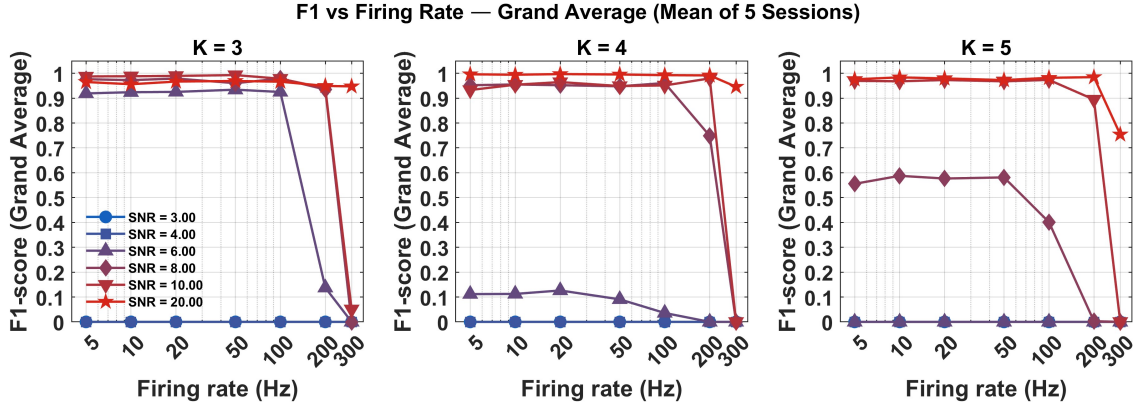


Figure 8.2. Grand-average F_1 -score as a function of the firing rate (f_r), for each K_{Quiroga} value; each curve corresponds to one SNR level.

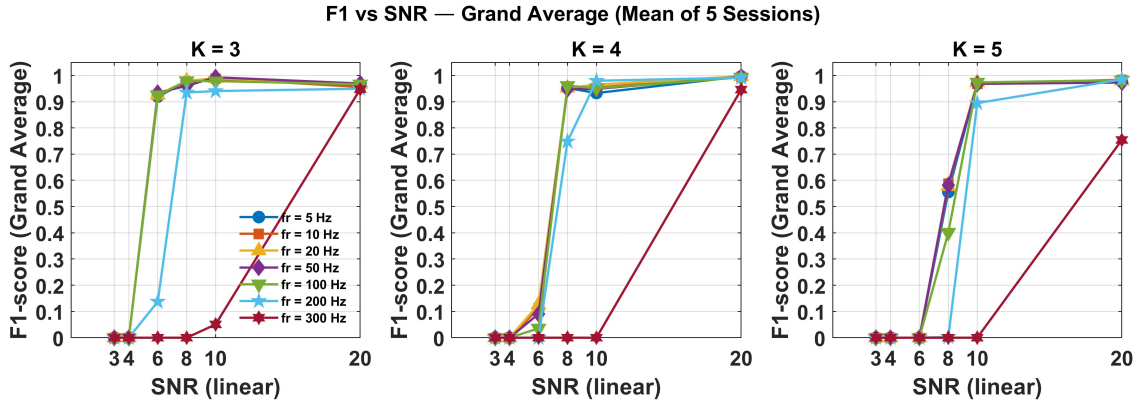


Figure 8.3. Grand-average F_1 -score as a function of the SNR, for each K_{Quiroga} value; each curve corresponds to one firing rate (f_r).

Table 8.2. Detection performance for $K_{\text{Quiroga}} = 3$: mean $\pm$ std of F_1 , precision, and recall across the SNR- f_r space.

SNR	f_r (Hz)	F_1	Precision	Recall
6	5	0.919 ± 0.131	0.884 ± 0.017	0.963 ± 0.011
6	10	0.924 ± 0.127	0.893 ± 0.028	0.967 ± 0.007
6	20	0.925 ± 0.128	0.899 ± 0.012	0.962 ± 0.005
6	50	0.934 ± 0.118	0.919 ± 0.015	0.955 ± 0.015
6	100	0.925 ± 0.132	0.945 ± 0.009	0.912 ± 0.018
6	200	0.138 ± 0.202	0.772 ± 0.087	0.090 ± 0.027
6	300	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
8	5	0.976 ± 0.129	0.978 ± 0.017	0.976 ± 0.014
8	10	0.973 ± 0.130	0.972 ± 0.010	0.973 ± 0.011
8	20	0.979 ± 0.122	0.987 ± 0.013	0.979 ± 0.011
8	50	0.960 ± 0.127	0.962 ± 0.013	0.960 ± 0.015
8	100	0.979 ± 0.102	0.979 ± 0.004	0.979 ± 0.004
8	200	0.935 ± 0.118	0.936 ± 0.037	0.935 ± 0.038
8	300	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
10	5	0.987 ± 0.085	0.988 ± 0.011	0.987 ± 0.010
10	10	0.988 ± 0.103	0.988 ± 0.008	0.988 ± 0.007
10	20	0.990 ± 0.059	0.994 ± 0.014	0.989 ± 0.013
10	50	0.993 ± 0.071	0.998 ± 0.003	0.992 ± 0.007
10	100	0.978 ± 0.086	0.979 ± 0.014	0.978 ± 0.014
10	200	0.940 ± 0.215	0.955 ± 0.057	0.938 ± 0.067
10	300	0.051 ± 0.182	0.165 ± 0.155	0.041 ± 0.038
20	5	0.965 ± 0.074	0.997 ± 0.004	0.941 ± 0.034
20	10	0.956 ± 0.058	0.984 ± 0.020	0.933 ± 0.043
20	20	0.967 ± 0.056	0.997 ± 0.004	0.943 ± 0.041
20	50	0.969 ± 0.051	0.997 ± 0.003	0.946 ± 0.035
20	100	0.966 ± 0.049	0.997 ± 0.005	0.941 ± 0.029
20	200	0.949 ± 0.116	0.991 ± 0.006	0.919 ± 0.028
20	300	0.947 ± 0.150	0.948 ± 0.078	0.947 ± 0.079

Table 8.3. Detection performance for $K_{\text{Quiroga}} = 4$: mean $\pm$ std of F_1 , precision, and recall across the SNR- f_r space.

SNR	f_r (Hz)	F_1	Precision	Recall
6	5	0.112 ± 0.113	0.686 ± 0.233	0.063 ± 0.054
6	10	0.113 ± 0.099	0.891 ± 0.087	0.063 ± 0.041
6	20	0.126 ± 0.113	0.922 ± 0.068	0.072 ± 0.053
6	50	0.091 ± 0.084	0.927 ± 0.051	0.050 ± 0.027
6	100	0.036 ± 0.060	0.834 ± 0.091	0.019 ± 0.019
6	200	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
6	300	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
8	5	0.952 ± 0.128	0.932 ± 0.025	0.979 ± 0.001
8	10	0.955 ± 0.131	0.934 ± 0.017	0.979 ± 0.002
8	20	0.952 ± 0.132	0.933 ± 0.019	0.974 ± 0.006
8	50	0.948 ± 0.157	0.943 ± 0.013	0.963 ± 0.022
8	100	0.961 ± 0.132	0.957 ± 0.007	0.966 ± 0.010
8	200	0.748 ± 0.275	0.982 ± 0.002	0.662 ± 0.083
8	300	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
10	5	0.933 ± 0.206	0.934 ± 0.059	0.934 ± 0.058
10	10	0.954 ± 0.135	0.957 ± 0.029	0.955 ± 0.025
10	20	0.965 ± 0.118	0.971 ± 0.030	0.965 ± 0.029
10	50	0.949 ± 0.163	0.954 ± 0.027	0.949 ± 0.029
10	100	0.951 ± 0.125	0.956 ± 0.018	0.951 ± 0.018
10	200	0.980 ± 0.103	0.979 ± 0.024	0.980 ± 0.024
10	300	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
20	5	0.996 ± 0.028	0.997 ± 0.003	0.996 ± 0.004
20	10	0.994 ± 0.042	0.995 ± 0.004	0.994 ± 0.004
20	20	0.997 ± 0.011	0.998 ± 0.003	0.996 ± 0.004
20	50	0.995 ± 0.058	0.999 ± 0.001	0.995 ± 0.007
20	100	0.993 ± 0.059	0.995 ± 0.006	0.992 ± 0.008
20	200	0.992 ± 0.083	0.997 ± 0.006	0.990 ± 0.009
20	300	0.946 ± 0.165	0.952 ± 0.080	0.946 ± 0.086

Table 8.4. Detection performance for $K_{\text{Quiroga}} = 5$: mean $\pm$ std of F_1 , precision, and recall across the SNR- f_r space.

SNR	f_r (Hz)	F_1	Precision	Recall
6	5	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000
6	10	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000
6	20	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000
6	50	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000
6	100	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000
6	200	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000
6	300	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000
8	5	0.555 $\pm$ 0.219	0.971 $\pm$ 0.017	0.416 $\pm$ 0.127
8	10	0.588 $\pm$ 0.236	0.973 $\pm$ 0.021	0.455 $\pm$ 0.155
8	20	0.577 $\pm$ 0.225	0.967 $\pm$ 0.018	0.445 $\pm$ 0.135
8	50	0.581 $\pm$ 0.250	0.982 $\pm$ 0.002	0.451 $\pm$ 0.130
8	100	0.401 $\pm$ 0.264	0.979 $\pm$ 0.008	0.287 $\pm$ 0.112
8	200	0.003 $\pm$ 0.015	0.131 $\pm$ 0.129	0.001 $\pm$ 0.002
8	300	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000
10	5	0.970 $\pm$ 0.130	0.958 $\pm$ 0.009	0.982 $\pm$ 0.001
10	10	0.968 $\pm$ 0.130	0.955 $\pm$ 0.011	0.981 $\pm$ 0.002
10	20	0.973 $\pm$ 0.118	0.965 $\pm$ 0.009	0.982 $\pm$ 0.004
10	50	0.968 $\pm$ 0.132	0.960 $\pm$ 0.009	0.976 $\pm$ 0.005
10	100	0.974 $\pm$ 0.130	0.970 $\pm$ 0.005	0.977 $\pm$ 0.002
10	200	0.895 $\pm$ 0.201	0.982 $\pm$ 0.001	0.852 $\pm$ 0.058
10	300	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000	0.000 $\pm$ 0.000
20	5	0.976 $\pm$ 0.130	0.976 $\pm$ 0.007	0.976 $\pm$ 0.007
20	10	0.984 $\pm$ 0.104	0.986 $\pm$ 0.005	0.983 $\pm$ 0.002
20	20	0.979 $\pm$ 0.112	0.982 $\pm$ 0.015	0.979 $\pm$ 0.013
20	50	0.973 $\pm$ 0.151	0.976 $\pm$ 0.009	0.973 $\pm$ 0.015
20	100	0.982 $\pm$ 0.130	0.989 $\pm$ 0.010	0.982 $\pm$ 0.022
20	200	0.985 $\pm$ 0.118	0.987 $\pm$ 0.014	0.984 $\pm$ 0.019
20	300	0.754 $\pm$ 0.373	0.926 $\pm$ 0.078	0.732 $\pm$ 0.141

8.1.3 Statistical Significance and Effect Size

Figure 8.4 reports the F_1 -score across the SNR- f_r space for each K_{Quiroga} value; white asterisks mark the conditions in which that K_{Quiroga} value yields a statistically significant optimum, based on the Friedman test described in Section 7.2, after correction for the false discovery rate (FDR). All 42 experimental conditions were tested, as each was characterised by at least three common sessions. Of these, in 20 the effect of K_{Quiroga} was found to be statistically significant after FDR correction ($\alpha = 0.05$).

Moreover, among the 20 significant conditions, $K_{\text{Quiroga}} = 3$ is the optimal value in 16 cases, while $K_{\text{Quiroga}} = 4$ is optimal in the remaining 4. $K_{\text{Quiroga}} = 5$ was never optimal among the significant conditions. In particular, $K_{\text{Quiroga}} = 3$ prevails at SNR = 6, SNR = 8, and SNR = 10. $K_{\text{Quiroga}} = 4$ is predominant at SNR = 20, except at the highest firing rate ($f_r = 300$ Hz), where $K_{\text{Quiroga}} = 3$ again becomes optimal.

Among the 20 statistically significant conditions, Kendall's W ranges from 0.760 to 1.000, with a mean of 0.900 and all 20 conditions (100%) correspond to a large effect ($W \geq 0.5$). The complete list of significant conditions, together with the corresponding p -values and Kendall's W , is given in Table 8.5.

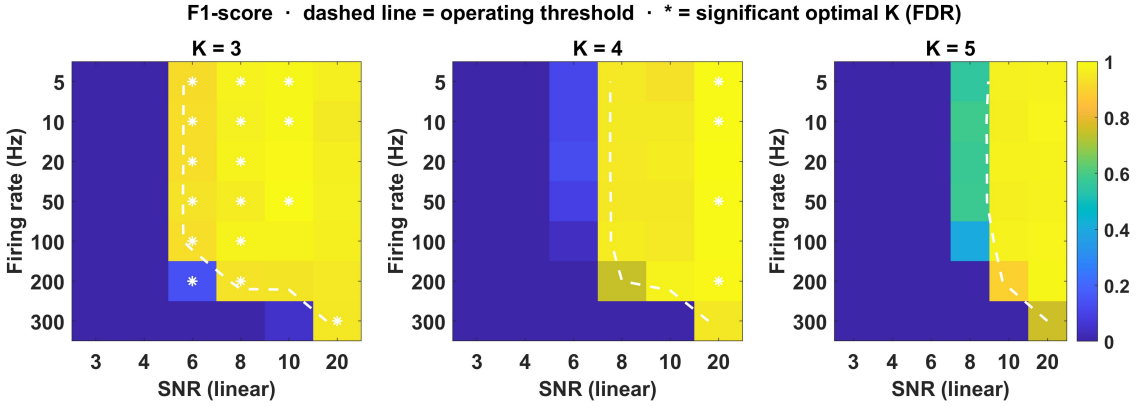


Figure 8.4. F_1 -score across the SNR- f_r space for each K_{Quiroga} value. White asterisks mark conditions in which that K_{Quiroga} value is the statistically significant optimum (Friedman test, FDR-corrected); the white dashed line marks the $F_1 \geq 0.75$ operating threshold.

Table 8.5. Conditions with a statistically significant effect of K_{Quiroga} on F_1 (Friedman test, FDR-corrected, $\alpha = 0.05$).

SNR	f_r (Hz)	Optimal K_{Quiroga}	p	Kendall's W
6	5	3	0.00674	1.00
6	10	3	0.00674	1.00
6	20	3	0.00674	1.00
6	50	3	0.00674	1.00
6	100	3	0.00674	1.00
6	200	3	0.00674	1.00
8	5	3	0.01500	0.84
8	10	3	0.00674	1.00
8	20	3	0.01500	0.84
8	50	3	0.01500	0.84
8	100	3	0.00674	1.00
8	200	3	0.00674	1.00
10	5	3	0.02237	0.76
10	10	3	0.01500	0.84
10	50	3	0.01500	0.84
20	5	4	0.02237	0.76
20	10	4	0.00674	1.00
20	50	4	0.02237	0.76
20	200	4	0.02237	0.76
20	300	3	0.02237	0.76

8.1.4 Temporal Accuracy

Temporal jitter was evaluated as a function of SNR for each value of K_{Quiroga} . In this case, the analysis was restricted to firing rates up to 100 Hz and to conditions characterised by effective detection alone. The data obtained allow jitter to be estimated from SNR = 6 for $K_{\text{Quiroga}} = 3$, whereas for $K_{\text{Quiroga}} = 4$ and $K_{\text{Quiroga}} = 5$ the estimate is possible only from the SNR = 8 condition onward.

Figure 8.5 reports the mean jitter as a function of SNR for each value of K_{Quiroga} , while the corresponding mean $\pm$ standard deviation values are reported numerically in Table 8.6. For $K_{\text{Quiroga}} = 3$, jitter ranges from 0.175 ms at SNR = 20 to 0.367 ms at SNR = 8. For $K_{\text{Quiroga}} = 4$, it ranges from 0.168 ms to 0.320 ms, while for $K_{\text{Quiroga}} = 5$ it takes values between 0.143 ms and 0.237 ms. Figure 8.6 additionally reports the pooled jitter distribution for each value of K_{Quiroga} , considering all conditions with SNR ≥ 8 that exhibit effective detection.

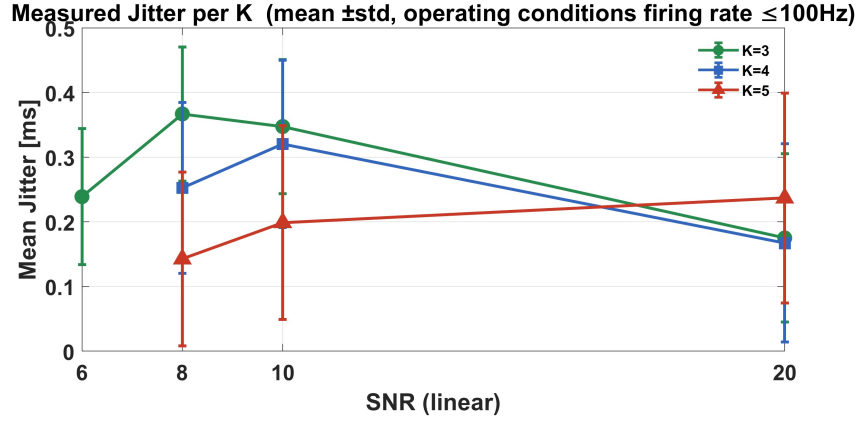


Figure 8.5. Mean jitter ($\pm$ std) as a function of SNR, for each K_{Quiroga} value, restricted to firing rates up to 100 Hz.

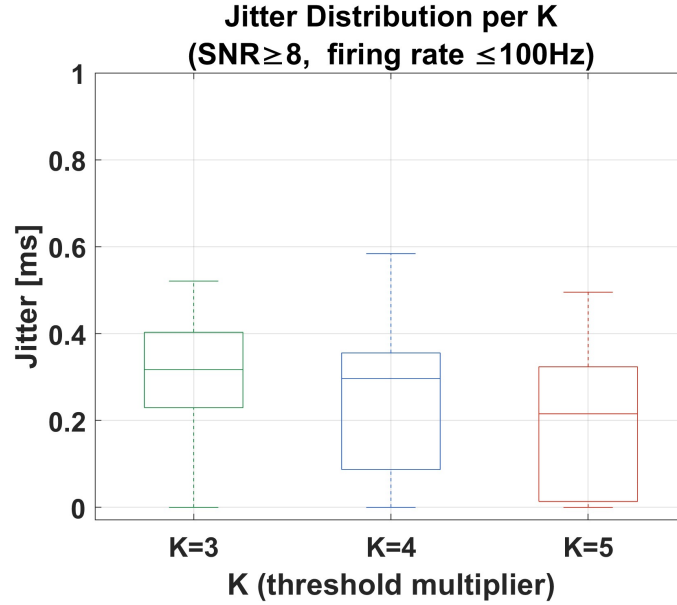


Figure 8.6. Pooled jitter distribution for each K_{Quiroga} value, across SNR ≥ 8 and firing rates up to 100 Hz.

Table 8.6. Mean jitter ($\pm$ std, in ms) per K_{Quiroga} and SNR, restricted to firing rates up to 100 Hz.

SNR	$K_{\text{Quiroga}} = 3$	$K_{\text{Quiroga}} = 4$	$K_{\text{Quiroga}} = 5$
6	0.239 ± 0.105	—	—
8	0.367 ± 0.104	0.252 ± 0.132	0.143 ± 0.134
10	0.347 ± 0.104	0.320 ± 0.129	0.199 ± 0.150
20	0.175 ± 0.130	0.168 ± 0.153	0.237 ± 0.163

8.2 Study 2 — Amplitude Sweep with Fixed Calibration

Study 2 evaluates a condition in which spike amplitude does not remain constant over time, reproducing the amplitude drift that a real culture can undergo during a session (Section 6.3.2). The study was conducted with K_{Quiroga} , SNR, and f_r fixed at the values established in Study 1 ($K_{\text{Quiroga}} = 4$, SNR = 10, and $f_r = 50$ Hz), and calibration is performed only once, at $A_{\text{ref}} = 150$ mV, producing a fixed noise estimate $\sigma_{\text{cal}} = 15$ mV and the corresponding DAC threshold, which is held constant across all five examined spike amplitudes. The SNR parameter, fixed at 10, is descriptive only of the first measurement, in which calibration also takes place; consequently, the use of a constant sigma affects the SNR value across the different conditions, which decreases as amplitude decreases, going from an optimal value of 10 to a critical value of 3.3.

Figure 8.7 reports the F_1 score as a function of spike amplitude for each of the five sessions, while Table 8.7 reports the mean $\pm$ standard deviation values for F_1 , precision, recall, and jitter. At $A_{\text{spike}} = 50$ mV (effective SNR = 3.33, below K_{Quiroga}), the F_1 score drops up to 0.570 ± 0.302 , with precision remaining high around 0.974 ± 0.119 , in contrast with recall decreasing to 0.461 ± 0.294 . Analyzing all the other acquisitions with higher peak amplitude, F_1 score increases markedly without an evident mismatch between precision and recall. Moreover, jitter varies between 0.232 ms and 0.396 ms across the five amplitude conditions, without showing a clear monotonic trend.

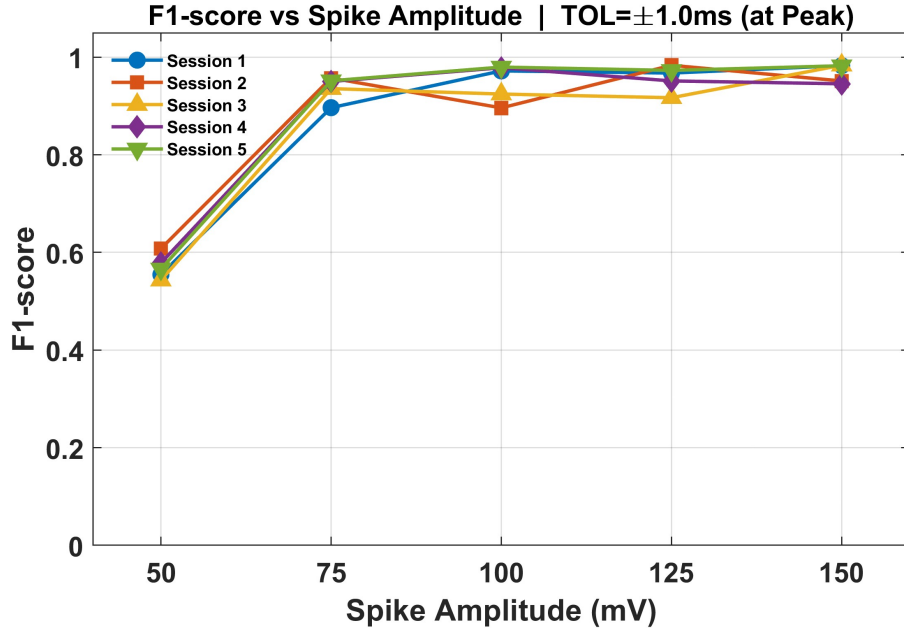


Figure 8.7. F_1 -score as a function of spike amplitude A_{spike} , shown separately for each of the 5 sessions.

Table 8.7. Detection performance and jitter as a function of spike amplitude A_{spike} (Study 2). Calibration was performed once at $A_{\text{ref}} = 150 \text{ mV}$ ($\text{SNR} = 10$, $\sigma_{\text{cal}} = 15 \text{ mV}$), and the resulting DAC threshold was held fixed across all five tested amplitudes; the effective SNR at test time is computed as $A_{\text{spike}}/\sigma_{\text{cal}}$.

A_{spike} (mV)	Effective SNR	F_1	Precision	Recall	Jitter (ms)
50	3.33	0.570 ± 0.302	0.974 ± 0.119	0.461 ± 0.294	0.232 ± 0.116
75	5.00	0.938 ± 0.131	0.962 ± 0.118	0.921 ± 0.146	0.396 ± 0.083
100	6.67	0.950 ± 0.123	0.951 ± 0.123	0.950 ± 0.124	0.349 ± 0.151
125	8.33	0.958 ± 0.120	0.958 ± 0.120	0.959 ± 0.119	0.371 ± 0.095
150	10.00	0.969 ± 0.117	0.977 ± 0.069	0.969 ± 0.118	0.366 ± 0.091

8.3 Study 3 — Poisson Firing Rate

Study 3 evaluates detection performance under a stochastic Poisson spike timing model, instead of the fixed inter-spike interval used in Studies 1 and 2, with K_{Quiroga} , SNR, and spike amplitude fixed at the values established in Study 1 ($K_{\text{Quiroga}} = 4$, $\text{SNR} = 10$, $A_{\text{spike}} = 150 \text{ mV}$). The mean firing rate is swept over the same seven values used in Study 1, from 5 to 300 Hz.

Figure 8.8 reports the F_1 score as a function of mean firing rate for each of the five sessions, while Table 8.8 reports the mean $\pm$ std values for F_1 , precision, recall, and jitter. A decrease in F_1 is evident in all sessions as the firing rate increases, dropping from 0.935 ± 0.117 at 5 Hz to 0.639 ± 0.130 at 300 Hz. The decrease in the F_1 score is directly associated with the reduction in recall; in fact, precision stays high across all firing rates (0.92–0.98), while recall declines steadily, from 0.930 at 5 Hz to 0.497 at 300 Hz. Jitter stays within a narrow range, between 0.344 ms and 0.432 ms, across all firing rates.

Table 8.8. Detection performance and jitter as a function of mean firing rate (Study 3, Poisson timing). $K_{\text{Quiroga}} = 4$, $\text{SNR} = 10$, $A_{\text{spike}} = 150 \text{ mV}$ fixed.

Mean f_r (Hz)	F_1	Precision	Recall	Jitter (ms)
5	0.935 ± 0.117	0.948 ± 0.084	0.930 ± 0.120	0.344 ± 0.024
10	0.933 ± 0.124	0.955 ± 0.085	0.922 ± 0.125	0.401 ± 0.026
20	0.927 ± 0.092	0.954 ± 0.062	0.907 ± 0.092	0.358 ± 0.037
50	0.903 ± 0.079	0.966 ± 0.025	0.853 ± 0.080	0.376 ± 0.034
100	0.855 ± 0.075	0.963 ± 0.034	0.773 ± 0.073	0.402 ± 0.024
200	0.779 ± 0.069	0.976 ± 0.023	0.652 ± 0.064	0.405 ± 0.033
300	0.639 ± 0.130	0.922 ± 0.091	0.497 ± 0.111	0.432 ± 0.039

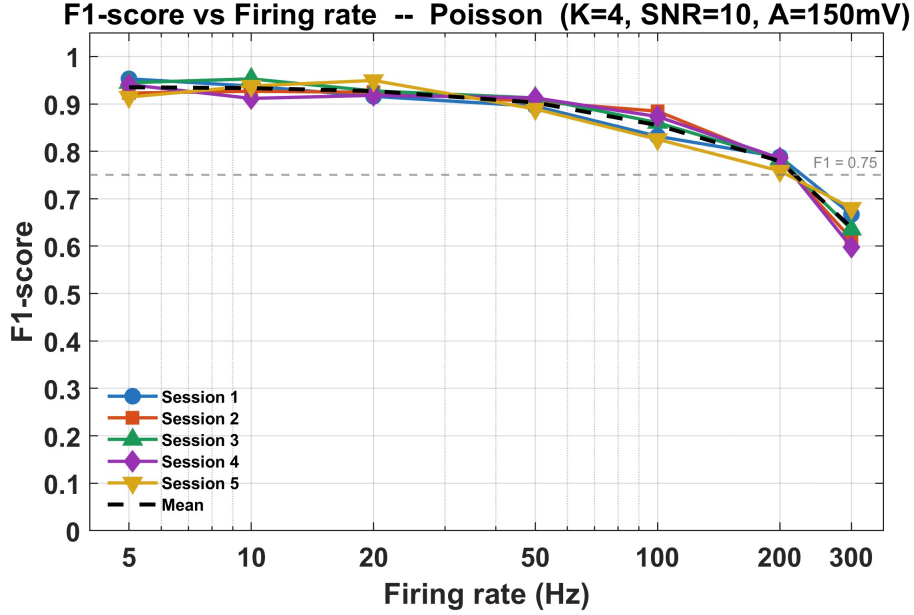


Figure 8.8. F_1 -score as a function of mean firing rate under Poisson spike timing, shown separately for each of the 5 sessions, with the mean across sessions (dashed black line).

8.4 Study 4 — Onboard Latency Characterization

Study 4 characterizes the onboard temporal latency of the acquisition pipeline through two complementary measurements: the FPGA-internal latency τ_{FPGA} , isolating the contribution of the digital detection logic, and the end-to-end system latency τ_{system} , which additionally includes the analog signal path (Section 6.3.4). The following sections report the descriptive statistics for both quantities, followed by the inferential analysis of the effect of SNR and K_{Quiroga} on τ_{system} .

8.4.1 FPGA-Internal Latency

The FPGA-internal latency τ_{FPGA} was measured over 15 independent sessions at fixed conditions ($\text{SNR} = 10$, $K_{\text{Quiroga}} = 4$), probing the `sticky_mask` output of channel 33 as described in Section 6.3.4. Table 8.9 reports the per-session mean, standard deviation, and number of examined edges.

Given the pipeline structure described in Section 4.5, where a minimum of 3 clock cycles for the majority-vote debouncer to assert `syncd` followed by one additional registered cycle for the `sticky_mask` update must occur, the theoretical latency range at the 50 MHz system clock is [80, 100) ns: the lower bound corresponds to a spike aligned with a clock edge, while the upper bound is approached as the spike occurrence drifts towards the end of a clock period. The measured grand-mean latency is 80.41 ± 3.39 ns (inter-session mean and standard deviation), and the within-session standard deviation remains below 20 ns in all 15 sessions.

Table 8.9. Per-session results for the FPGA-internal latency (τ_{FPGA}).

Session	Mean (ns)	Std (ns)	Edges examined
1	80.90	9.801	354
2	79.84	11.310	357
3	77.78	16.580	354
4	77.40	15.970	359
5	78.08	16.970	357
6	91.75	14.410	356
7	78.82	9.975	358
8	79.31	8.000	354
9	79.06	12.480	353
10	79.46	9.363	356
11	79.12	12.000	359
12	81.06	10.640	357
13	80.81	12.440	353
14	81.23	8.856	358
15	81.54	6.824	353
Grand mean	80.41 $\pm$ 3.39	12.09	—

8.4.2 System Latency — Descriptive Statistics

Table 8.10 and Table 8.11 report, for each of the ten sessions per SNR condition, the per-session mean, standard deviation, and median of the end-to-end system latency τ_{system} , together with the number of matched (DAQ peak, GPIO edge) pairs, for $K_{\text{Quiroga}} = 3$ and $K_{\text{Quiroga}} = 4$ respectively.

Across both K_{Quiroga} values, the session-level mean latency increases as SNR decreases: for $K_{\text{Quiroga}} = 3$, it ranges from 295–505 μs at SNR = 20, through 615–779 μs at SNR = 10, to 904–1194 μs at SNR = 8; for $K_{\text{Quiroga}} = 4$, the corresponding ranges are 269–435 μs , 882–1182 μs , and 1399–1508 μs , respectively. The session-level median is considerably more stable than the mean within each condition: at $K_{\text{Quiroga}} = 3$, SNR = 20, for instance, the median remains fixed at 300 μs in all ten sessions, despite the mean varying between 295 and 505 μs .

The standard deviation does not follow a monotonic trend with SNR: at SNR = 20 it spans a wide range (57–795 μs for $K_{\text{Quiroga}} = 3$; 59–601 μs for $K_{\text{Quiroga}} = 4$), comparable to, or in some sessions larger than, the range observed at SNR = 8 (145–606 μs for $K_{\text{Quiroga}} = 3$; 84–543 μs for $K_{\text{Quiroga}} = 4$).

Table 8.10. Per-session descriptive statistics of the system latency (τ_{system}) for $K_{\text{Quiroga}} = 3$.

SNR	Session	Mean (μs)	$\sigma_{\tau_{\text{system}}} (\mu\text{s})$	Median (μs)	Matched events
20	1	324	270	300	499
20	2	305	63	300	500
20	3	361	380	300	499
20	4	350	327	300	499
20	5	314	337	300	497
20	6	309	65	300	499
20	7	295	57	300	494
20	8	297	190	300	467
20	9	372	529	300	498
20	10	505	795	300	499
Grand mean		343.2	373.8	—	—
10	1	740	465	700	496
10	2	615	193	630	492
10	3	716	364	700	499
10	4	641	159	700	499
10	5	710	594	670	495
10	6	694	178	700	488
10	7	686	133	700	499
10	8	716	551	700	499
10	9	717	441	700	497
10	10	779	506	700	499
Grand mean		701.4	395.9	—	—
8	1	1143	326	1100	495
8	2	927	163	930	499
8	3	1001	226	1000	499
8	4	937	161	910	499
8	5	1007	366	1000	499
8	6	941	176	1000	483
8	7	1033	400	1000	499
8	8	904	192	905	499
8	9	1194	606	1100	499
8	10	965	145	1000	499
Grand mean		1005.2	309.8	—	—

Table 8.11. Per-session descriptive statistics of the system latency (τ_{system}) for $K_{\text{Quiroga}} = 4$.

SNR	Session	Mean (μs)	$\sigma_{\tau_{\text{system}}} (\mu\text{s})$	Median (μs)	Matched events
20	1	269	68	300	500
20	2	334	464	300	478
20	3	354	406	300	498
20	4	339	318	300	496
20	5	309	79	300	500
20	6	375	86	400	499
20	7	385	341	300	498
20	8	304	59	300	499
20	9	367	298	340	498
20	10	435	601	320	499
Grand mean		347.1	327.0	—	—
10	1	1182	481	1100	495
10	2	1044	405	1000	498
10	3	1089	122	1100	499
10	4	975	163	1000	499
10	5	882	296	1000	500
10	6	1092	359	1070	499
10	7	1044	153	1060	499
10	8	1008	103	1000	499
10	9	1039	603	940	499
10	10	1071	540	1000	495
Grand mean		1042.6	366.4	—	—
8	1	1458	84	1485	440
8	2	1399	297	1400	502
8	3	1428	101	1400	496
8	4	1421	84	1400	497
8	5	1421	86	1400	484
8	6	1446	209	1400	494
8	7	1442	88	1420	483
8	8	1499	543	1400	500
8	9	1430	87	1400	484
8	10	1508	470	1400	488
Grand mean		1445.2	263.6	—	—

8.4.3 System Latency — Effect of SNR (Friedman and Kendall’s W)

For each value of K_{Quiroga} , the ten session medians of τ_{system} across the three SNR levels were submitted to the Friedman test, following the procedure described in Section 7.2.2. Table 8.12 reports the resulting test statistics. Both conditions reach statistical significance after Benjamini–Hochberg correction, with Kendall’s $W = 1.000$ in both cases, indicating perfect rank concordance across all ten sessions.

Table 8.12. Friedman test results for the effect of SNR on the session-level median of τ_{system} , per K_{Quiroga} value (BH-FDR corrected, $\alpha = 0.05$, $m = 2$).

K_{Quiroga}	χ^2_{Friedman}	p	Kendall’s W	Outcome (BH-FDR)
3	20.000	< 0.0001	1.000	Significant*
4	20.000	< 0.0001	1.000	Significant*

Since the Friedman test is an overall test, post-hoc pairwise comparisons were performed for the three SNR pairs, separately for each K_{Quiroga} value, using the exact Wilcoxon signed-rank test on the paired session medians (Section 7.2.2). Figure 8.9 shows the median latency for each condition, with significance brackets marking the outcome of each pairwise comparison. All six comparisons — three SNR pairs for each of the two K_{Quiroga} values — are significant after BH-FDR correction ($m = 6$, $p = 0.002$ in every case), with a rank-biserial effect size $r = 1.000$ throughout, indicating that the direction of the difference is fully consistent across all ten sessions in every comparison.

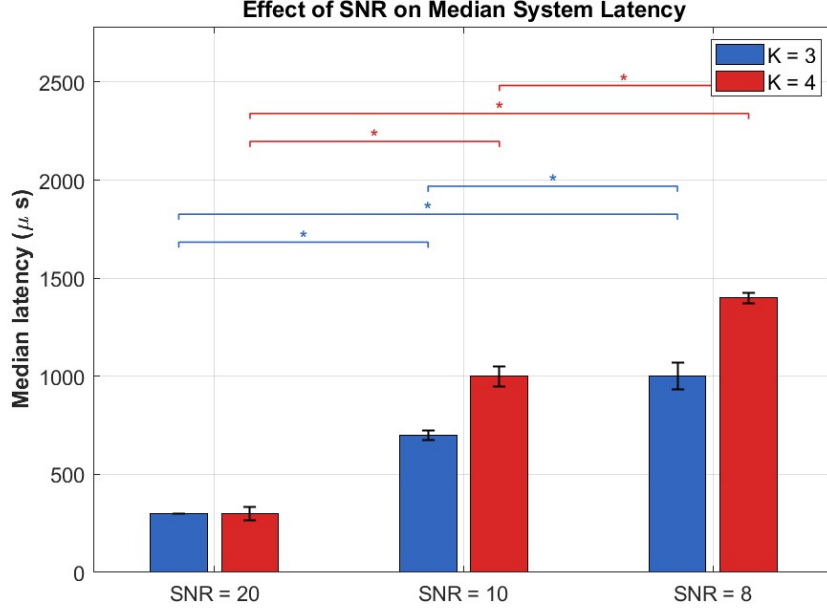


Figure 8.9. Median system latency (τ_{system}) by SNR and K_{Quiroga} , with error bars showing the standard deviation across the ten session medians. Brackets mark the outcome of the post-hoc pairwise Wilcoxon signed-rank tests (BH-FDR corrected); * denotes a statistically significant difference.

8.4.4 System Latency — Effect of K_{Quiroga} (Wilcoxon signed-rank)

For each SNR level, the ten paired session medians of τ_{system} for $K_{\text{Quiroga}} = 3$ and $K_{\text{Quiroga}} = 4$ were compared using the exact Wilcoxon signed-rank test, following the procedure described in Section 7.2.2. Figure 8.10 shows the median latency for both K_{Quiroga} values at each SNR level, with a significance bracket marking the outcome of the paired comparison; Table 8.13 reports the corresponding numerical results.

The outcome is not uniform across conditions: at SNR = 20, the difference between $K_{\text{Quiroga}} = 3$ and $K_{\text{Quiroga}} = 4$ is not statistically significant after BH-FDR correction ($p = 0.238$, $r = 0.564$). At SNR = 10 and SNR = 8, by contrast, the difference is significant ($p = 0.002$ in both cases), with a rank-biserial effect size $r = 1.000$, indicating that $K_{\text{Quiroga}} = 3$ yields a lower median latency than $K_{\text{Quiroga}} = 4$ in all ten sessions at both SNR levels.

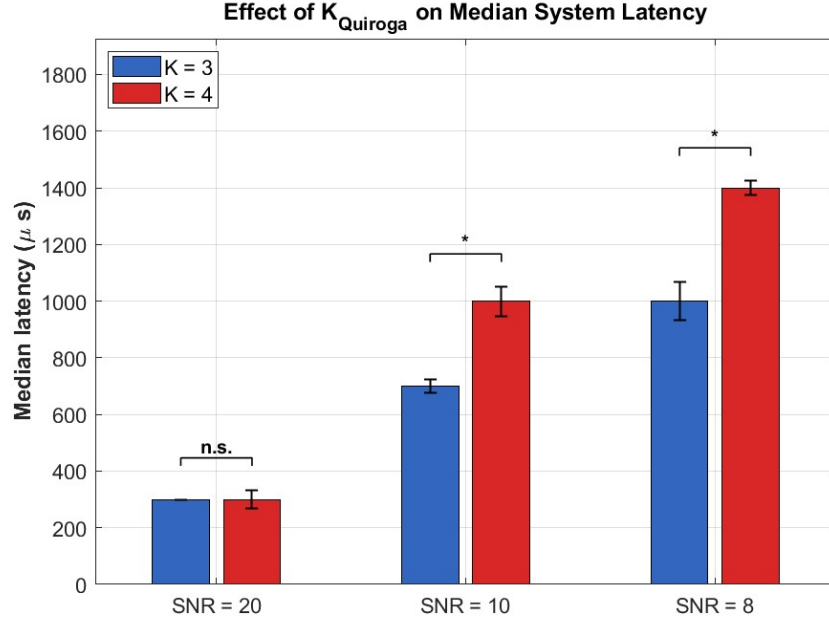


Figure 8.10. Median system latency (τ_{system}) by K_{Quiroga} and SNR, with error bars showing the standard deviation across the ten session medians. Brackets mark the outcome of the exact Wilcoxon signed-rank test for $K_{\text{Quiroga}} = 3$ vs $K_{\text{Quiroga}} = 4$ (BH-FDR corrected); * denotes a statistically significant difference, **n.s.** a non-significant one.

Table 8.13. Wilcoxon signed-rank test results (exact method) for the effect of K_{Quiroga} on the session-level median of τ_{system} , per SNR level (BH-FDR corrected, $\alpha = 0.05$, $m = 3$).

SNR	p	r	Outcome (BH-FDR)
20	0.238	0.564	n.s.
10	0.002	1.000	Significant*
8	0.002	1.000	Significant*

Chapter 9

Discussion

The following chapter discusses and interprets the results obtained in the four experimental studies described in Chapter 6 and reported in Chapter 8.

The discussion is organised around the K/SNR ratio, which fixes the position of the detection threshold relative to the spike peak and so governs whether an event is detected at all. This ratio explains two different ways in which the system fails, one driven by insufficient SNR and one by high firing rate, and it also underlies the choice of the optimal K value and the resulting operating range obtained from Study 1. The same framework guides the two robustness analyses: Study 2, compared against Study 1 at matched SNR, shows how much calibration quality alone determines performance, while Study 3 tests whether realistic Poisson timing changes the picture obtained under regular timing. Study 4 then treats the temporal behaviour of the platform, separating the negligible digital contribution from the analog-dominated end-to-end latency and comparing the result with values reported for similar systems. The chapter ends with the limitations of the work and possible future developments.

9.1 The K/SNR Ratio

Since the threshold is defined as $\text{Th} = K_{\text{Quiroga}} \cdot \sigma_{\text{noise}}$ and the signal-to-noise ratio as $\text{SNR} = A_{\text{spike}}/\sigma_{\text{noise}}$, the parameter that intrinsically determines the ability to detect an event is the ratio between the programmed detection threshold (Th) and the peak amplitude of the neural spike (A_{spike}):

$$\frac{\text{Th}}{A_{\text{spike}}} = \frac{K_{\text{Quiroga}} \cdot \sigma_{\text{noise}}}{A_{\text{spike}}} = \frac{K_{\text{Quiroga}}}{\text{SNR}} \quad (9.1)$$

Mapping the overall F_1 score obtained in Study 1 as a function of the $K_{\text{Quiroga}}/\text{SNR}$ ratio (Figure 9.1), the system behaviour can be articulated into three distinct operating zones, each governed by a different physical constraint:

- **High-performance zone** ($K_{\text{Quiroga}}/\text{SNR} < 0.5$): The threshold lies below half the spike peak amplitude, with a wide safety margin against stochastic noise fluctuations.

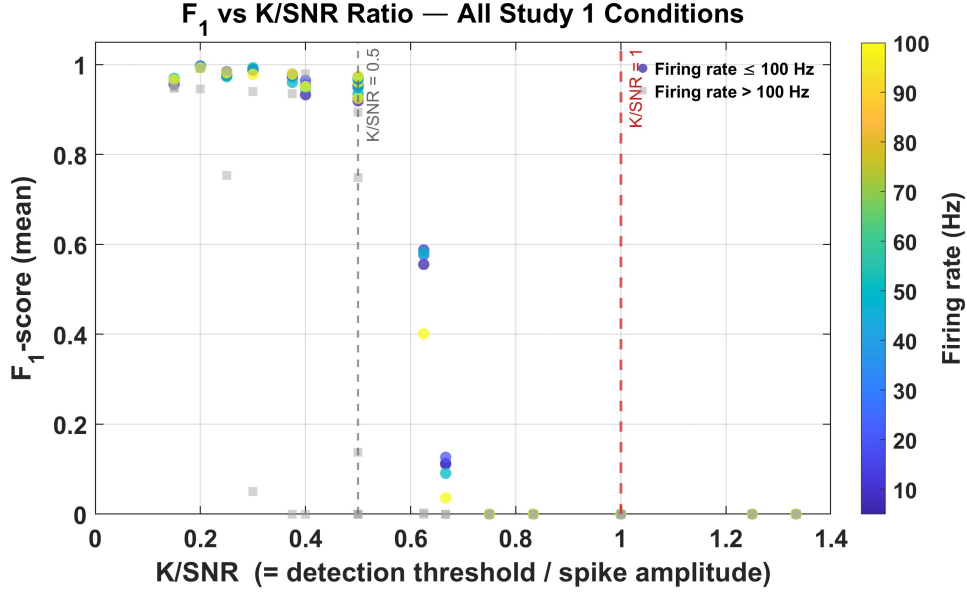


Figure 9.1. F_1 -score as a function of the $K_{\text{Quiroga}}/\text{SNR}$ ratio, aggregating all Study 1 conditions. Coloured circles correspond to firing rates $f_r \leq 100$ Hz (colour scale indicates f_r value); grey squares correspond to $f_r > 100$ Hz. The dashed grey vertical line marks $K/\text{SNR} = 0.5$; the dashed red line marks $K/\text{SNR} = 1.0$.

- **Rapid degradation zone** ($0.5 \lesssim K_{\text{Quiroga}}/\text{SNR} \lesssim 0.7$): The threshold approaches the spike peak, as shown in Figure 9.2 ($K_{\text{Quiroga}} = 4$, $\text{SNR} = 6$, $K/\text{SNR} = 0.67$), where the threshold sits at the top of the spike peaks and amplitude variations or noise fluctuations frequently prevent threshold crossing, causing a sharp Recall drop while Precision remains high, as documented in Tables 8.2–8.4.
- **Structural impossibility zone** ($K_{\text{Quiroga}}/\text{SNR} \geq 1.0$): The threshold is mathematically greater than or equal to the spike amplitude ($\text{Th} \geq A_{\text{spike}}$), as shown in Figure 9.3 ($K_{\text{Quiroga}} = 4$, $\text{SNR} = 3$), where the threshold lies well above the entire signal excursion, precluding detection by construction and yielding $F_1 = 0$.

Even though the pattern seems repetitive, the transition zone between 0.5 and 0.7 is covered by only two parameter combinations within Study 1, so the rapid collapse described in this interval should be regarded as a suggestive pattern indicating the operational limits of the system, rather than as a quantitatively established empirical law.

A second distinct failure mechanism occurs at high firing rates independently of the nominal SNR value, as shown in Figure 9.4 ($K_{\text{Quiroga}} = 4$, $\text{SNR} = 10$, $f_r = 300$ Hz), where the primary cause lies in the compromise of the calibration phase itself. In fact, at high firing rates, the 100 ms calibration window is objectively contaminated by the spikes, occupying around 60% of the window at 300 Hz, causing an evident drop in performance even using the hybrid noise estimation algorithm described in Section 5.2. The described pattern is probably caused by the fact that the P25 pre-estimator is already significantly shifted upward relative to the true background noise level, consequently raising the masking

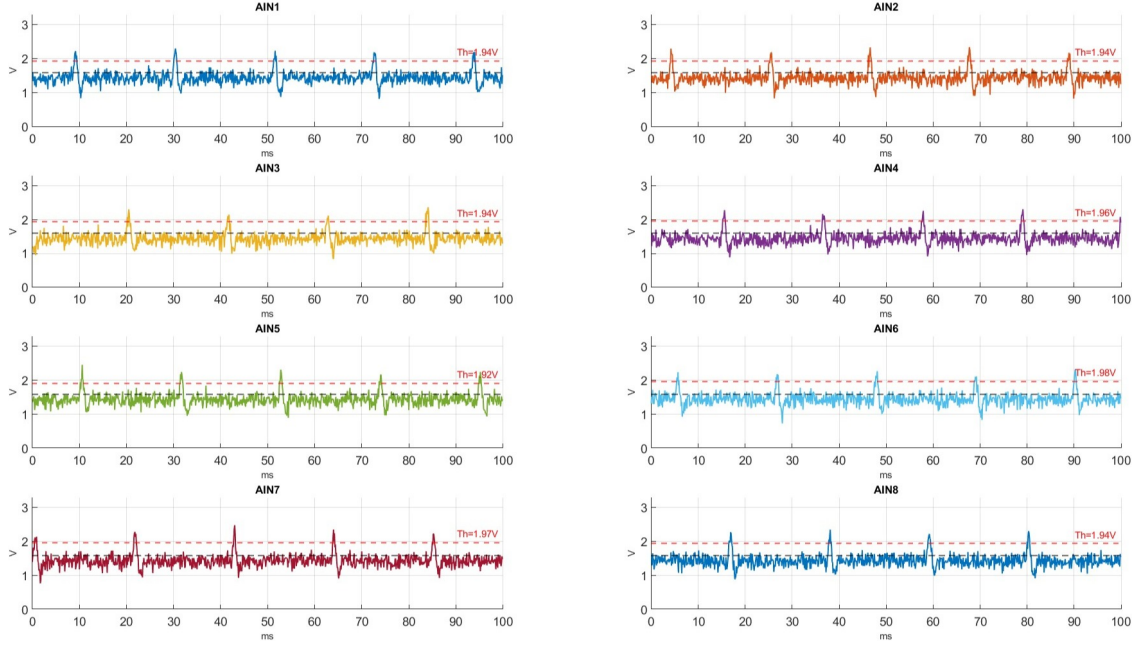


Figure 9.2. Representative calibration traces at $K_{\text{Quiroga}} = 4$, $\text{SNR} = 6$ ($K/\text{SNR} = 0.67$, transition zone). The detection threshold (red dashed line) sits at the top of the spike peaks, illustrating the fragile detection margin in this regime.

threshold ($4 \cdot \sigma_{\text{perc}}$) and reducing the number of spikes identified and removed. The net result is a DAC threshold that exceeds the spike amplitude during acquisition, effectively turning the case into one that is structurally analogous to $K_{\text{Quiroga}}/\text{SNR} \geq 1$, despite starting from nominally favourable parameters. The characteristic signature of this f_r -limited regime, in contrast to the SNR-limited case, is the simultaneous and symmetric collapse of both Precision and Recall, as observed for example at $K_{\text{Quiroga}} = 5$, $\text{SNR} = 8$, $f_r = 200$ Hz, where $F_1 = 0.003$, Precision = 0.131, and Recall = 0.001 (Table 8.4).

This interpretation is supported by the behaviour observed at $\text{SNR} = 20$, where the system recovers high performance even at $f_r = 300$ Hz ($F_1 = 0.947$ for $K_{\text{Quiroga}} = 3$, $F_1 = 0.946$ for $K_{\text{Quiroga}} = 4$; Tables 8.2 and 8.3): a proportionally higher spike amplitude ensures that the calibration-inflated threshold remains below the spike peak, restoring effective detection conditions.

9.2 Optimal K Selection and Operational Range

The statistical analysis of Study 1 (Section 8.1.3) identifies $K_{\text{Quiroga}} = 3$ as the optimal value in 16 of the 20 FDR-significant conditions, across $\text{SNR} = 6, 8$, and 10, whilst $K_{\text{Quiroga}} = 4$ becomes preferable only at $\text{SNR} = 20$, with the exception of $f_r = 300$ Hz and $K_{\text{Quiroga}} = 5$ is never optimal among the significant conditions.

The robustness of this recommendation is supported by the effect size analysis too, where all 20 FDR-significant conditions exhibit a large Kendall's concordance coefficient

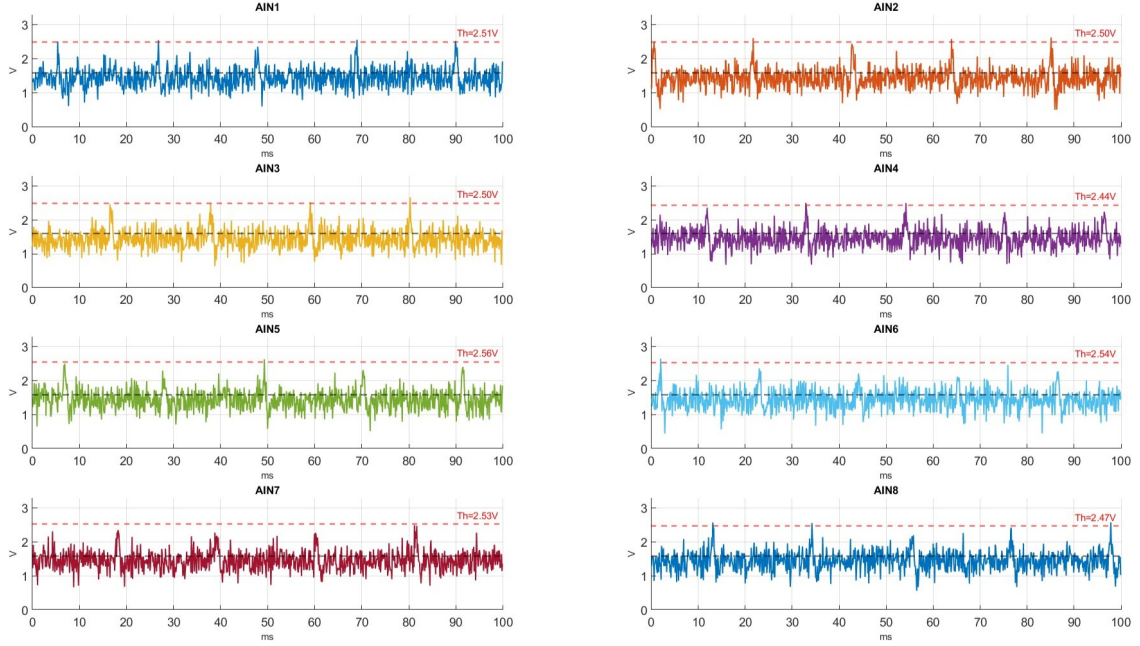


Figure 9.3. Representative calibration traces at $f_r = 50$ Hz, $K_{\text{Quiroga}} = 4$, $\text{SNR} = 3$ ($K/\text{SNR} = 1.33$, structural impossibility zone). The detection threshold lies well above the signal excursion, precluding detection by construction.

($W \geq 0.5$, mean $W = 0.900$), indicating that the session-to-session ranking of K values is highly consistent, proving that the preference for $K_{\text{Quiroga}} = 3$ is a stable property of the system across independent repetitions.

$K_{\text{Quiroga}} = 3$ therefore represents the recommended value for most practical operating conditions, where SNR is in the range 6–10.

Temporal jitter, reported in Section 8.1.4, exhibits a trend in which mean jitter decreases from 0.367 ms at $K_{\text{Quiroga}} = 3$ to 0.252 ms at $K_{\text{Quiroga}} = 4$ and 0.143 ms at $K_{\text{Quiroga}} = 5$, mostly due to a geometric interpretation where a higher K_{Quiroga} value places the detection threshold closer to the spike peak and therefore reduces the temporal window during which the signal exceeds the threshold, consequently reducing the temporal variability of the detected timestamp too.

This introduces a practical trade-off where $K_{\text{Quiroga}} = 3$ maximises detection sensitivity and F_1 across most operating conditions, but at the cost of a wider jitter, which is smallest at $K_{\text{Quiroga}} = 5$. $K_{\text{Quiroga}} = 4$ sits in between, retaining good detection performance while narrowing the jitter with respect to $K_{\text{Quiroga}} = 3$. This trend does not hold at high SNR , where the detection window is already narrow for all three K values and jitter no longer depends on the threshold geometry.

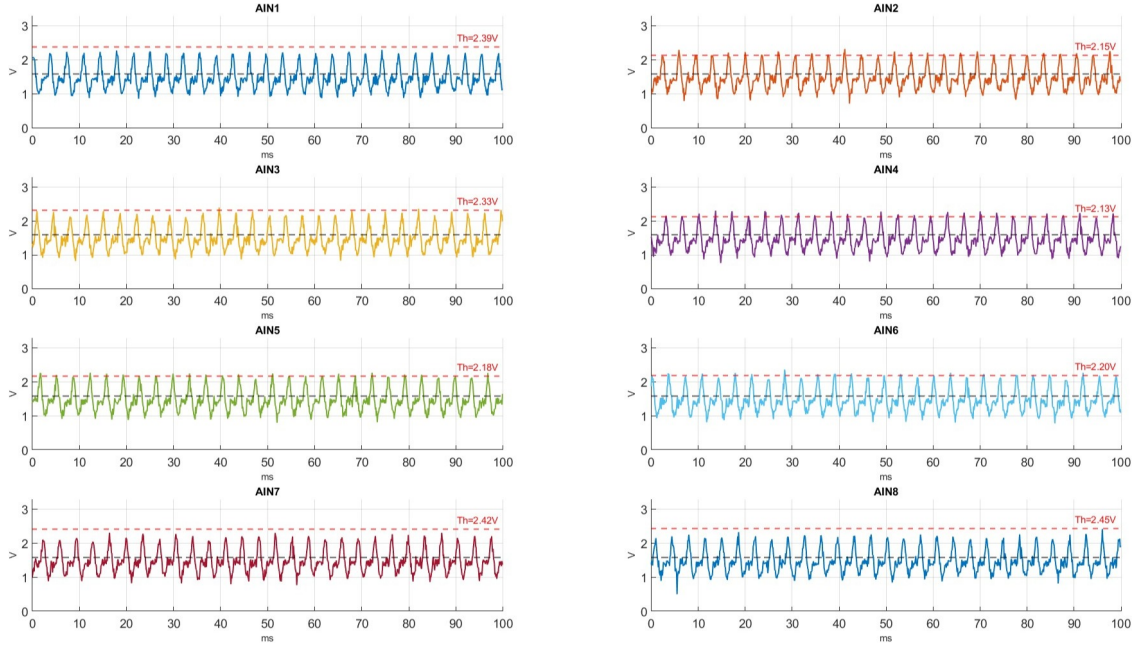


Figure 9.4. Representative calibration traces at $K_{\text{Quiroga}} = 4$, $\text{SNR} = 10$, $f_r = 300$ Hz. Despite a nominally favourable $K/\text{SNR} = 0.4$, the dense spike contamination of the calibration window causes the hybrid algorithm to overestimate σ_{noise} , producing a threshold that exceeds the spike amplitude.

9.3 Calibration Robustness: Study 1 vs Study 2

Study 2 was designed to evaluate the robustness of the detection pipeline when the spike amplitude at acquisition time differs from the amplitude present during calibration, simulating hypothetical electrode drift, neuronal adaptation, and coupling variations that are common in real recording sessions. The choice of $f_r = 50$ Hz reflects the upper range of spontaneous cortical firing rates observed in vitro, a condition that Study 1 consistently identifies as stable across all K values and free from f_r -limited failure.

The choice of $K_{\text{Quiroga}} = 4$ rests on three grounds:

- It corresponds to the canonical value proposed in the original Quiroga estimator [32], making the results directly comparable with the existing literature.
- At $\text{SNR} = 8$, $K_{\text{Quiroga}} = 4$ yields a ratio $K/\text{SNR} = 0.5$, which lies exactly at the boundary of the rapid degradation zone identified in Section 9.1.
- $K_{\text{Quiroga}} = 4$ represents a theoretical balance point between detection sensitivity and specificity, reducing the risk of false positives by setting a higher threshold and preserving recall by avoiding an excessively restrictive barrier.

The most important difference between Study 1 and Study 2 is evident in Figure 8.7 and Table 8.7: even at $A_{\text{spike}} = 50$ mV (effective $\text{SNR} = 3.33$, nominally below K_{Quiroga}), F_1 drops to 0.570 rather than collapsing to zero; precision remains high (0.974) while recall

falls to 0.461, indicating that the system continues to detect a meaningful fraction of events even under sub-threshold amplitude conditions.

The fundamental difference lies in calibration quality: in Study 1, the threshold is estimated at the degraded SNR itself, causing σ_{noise} to be overestimated, in contrast with Study 2, where calibration is performed once at $A_{\text{ref}} = 150 \text{ mV}$ and $\text{SNR} = 10$, yielding a fixed $\sigma_{\text{cal}} = 15 \text{ mV}$ and a well-anchored threshold; even when the spike amplitude subsequently decreases, this threshold remains a meaningful reference that the signal can still cross in the vicinity of the peak.

The practical implication is that calibration quality is a more critical determinant of detection performance than the instantaneous SNR at acquisition time. In a real recording session, where spike amplitudes may vary over time due to electrode drift or neuronal adaptation, performing a single reliable calibration at high SNR at session start provides substantially better robustness than recalibrating continuously under degraded signal conditions.

9.4 Robustness to Poisson Timing: Study 1 vs Study 3

Study 3 evaluates detection performance under Poisson-distributed spike timing, replacing the fixed inter-spike interval of Study 1 while keeping all other parameters identical ($K_{\text{Quiroga}} = 4$, $\text{SNR} = 10$, $A_{\text{spike}} = 150 \text{ mV}$).

The most significant comparison emerges at a mean firing rate of 300 Hz: in Study 1, $F_1 = 0.000$ for $K_{\text{Quiroga}} = 4$, $\text{SNR} = 10$, and $f_r = 300 \text{ Hz}$ (Table 8.3), in sharp contrast with Study 3, where Poisson timing at the same mean rate yields $F_1 = 0.639 \pm 0.130$ (Table 8.8).

The explanation lies in the calibration phase, and is well illustrated by comparing Figure 9.4 (300 Hz, fixed timing), which shows a uniformly dense spike train throughout the entire 100 ms calibration window with the resulting threshold ($\text{Th} \approx 2.1\text{--}2.4 \text{ V}$) systematically exceeding the spike peaks, with Figure 9.5 (300 Hz, Poisson timing, same mean rate), which instead shows a markedly lower threshold ($\text{Th} \approx 1.74\text{--}1.93 \text{ V}$). The reason lies primarily in the exponential distribution of inter-spike intervals, which produces visible periods of reduced activity interspersed among dense spike bursts, allowing the P25 pre-estimator to sample a sufficient proportion of background noise, limiting the overestimation of σ_{perc} and consequently producing a well-anchored threshold.

Even in Study 3, the system tends to favour precision over recall, consistent with a threshold that remains above the noise floor but is progressively less optimally positioned relative to the spike peaks, as calibration quality degrades with increasing spike density.

The irregular temporal structure of Poisson activity therefore partially protects the calibration phase from the contamination cascade described in Section 9.1, without eliminating it entirely, providing a more biologically representative estimate of system performance than Study 1 and suggesting that the abrupt performance collapse observed in Study 1 at high fixed firing rates is partly a consequence of the unrealistic regularity of the deterministic stimulus, and that under physiological conditions the performance degradation is instead more gradual and potentially recoverable.

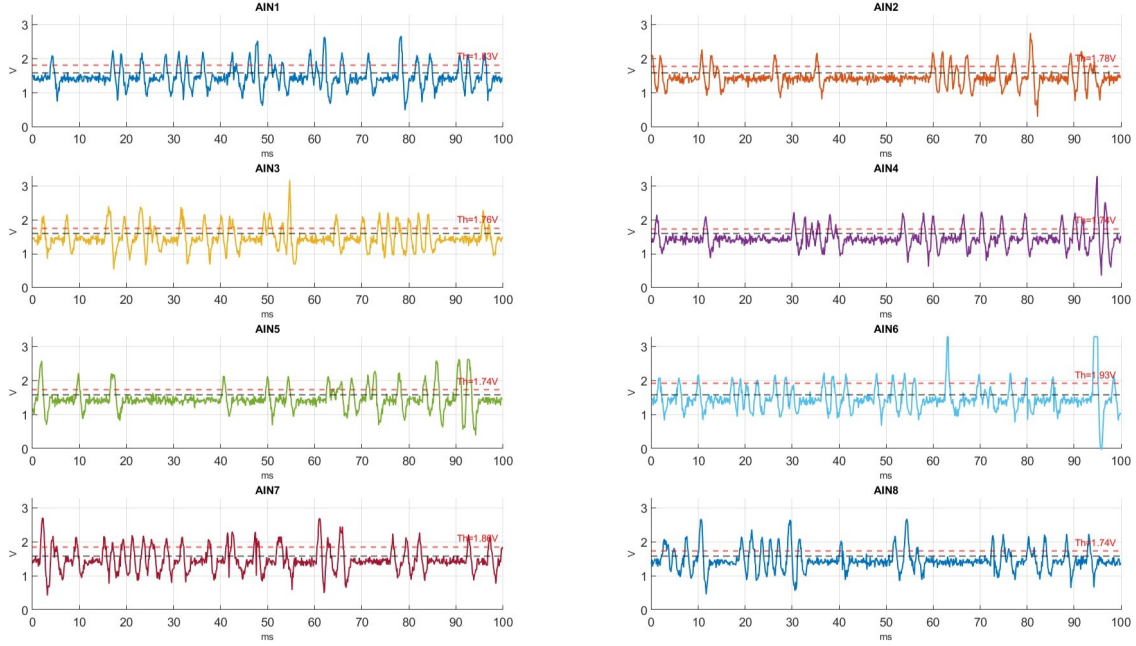


Figure 9.5. Representative calibration traces at $K_{\text{Quiroga}} = 4$, $\text{SNR} = 10$, mean $f_r = 300\text{ Hz}$ (Poisson timing). Compared to the fixed-timing case at the same parameters (Figure 9.4), the irregular spike distribution leaves visible periods of reduced activity, allowing the hybrid algorithm to estimate σ_{noise} more accurately. The resulting thresholds ($\text{Th} \approx 1.74\text{--}1.93\text{ V}$) fall below most spike peaks, restoring effective detection conditions despite the high mean firing rate.

9.5 System Latency Characterization: Study 4

Study 4 separates the acquisition pipeline into two temporal contributions in order to determine whether the latency overall is mainly dominated by one contribution or another, rather than treating it as a single aggregate figure.

The grand-mean $\tau_{\text{FPGA}} = 80.41 \pm 3.39\text{ ns}$ (Table 8.9, Section 8.4.1) falls within the theoretical range $[80, 100)\text{ ns}$ derived from the pipeline structure described in Section 4.5, proving that the RTL implementation behaves exactly as designed at the timing level. The grand mean standard deviation, which is only about 4% of the grand mean, points to high repeatability across independent sessions, while the within-session standard deviation, below 20 ns in all 15 sessions, shows that the digital stage adds essentially no temporal uncertainty of its own beyond the intrinsic quantization arising from the asynchronous arrival of the input edge relative to the sampling clock.

Comparing τ_{FPGA} with τ_{system} , its contribution is negligible in every condition: even in the most favourable case ($\text{SNR} = 20$, $K_{\text{Quiroga}} = 4$, as low as $269\text{ }\mu\text{s}$) it accounts for only about 0.03% of the total latency, falling to roughly 0.005% at the highest end-to-end latencies observed ($\text{SNR} = 8$, $K_{\text{Quiroga}} = 4$, up to $1508\text{ }\mu\text{s}$; Table 8.11). The digital detection logic is therefore negligible relative to the overall system response, confirming that almost the entire end-to-end latency budget is determined upstream of the FPGA, in

the analog signal path.

The effect of SNR on τ_{system} is evident in latency results, increasing it monotonically as SNR drops. For both K_{Quiroga} values (Friedman test, $\chi^2 = 20$, $p < 0.0001$, Kendall's $W = 1.000$; Table 8.12, Section 8.4.3, Figure 8.9), every session ranks the three SNR conditions in the same order, mostly due to the position of the calibrated threshold relative to the spike peak. Since $\text{Th} \propto K_{\text{Quiroga}} \cdot \sigma_{\text{noise}}$, a lower SNR corresponds to a larger σ_{noise} and consequently a higher threshold positioned closer to the spike peak amplitude. The closer the threshold sits to the peak, the later in the rising edge this crossing occurs, and the more time elapses between the DAQ-recorded peak and the assertion of the GPIO edge.

The effect of K_{Quiroga} follows the same logic, but its statistical significance depends strongly on the operating SNR (Table 8.13, Section 8.4.4, Figure 8.10). At SNR = 10 and SNR = 8, $K_{\text{Quiroga}} = 3$ shows a significantly lower latency than $K_{\text{Quiroga}} = 4$, whereas at SNR = 20 no significant difference is observed ($p = 0.238$, $r = 0.564$), confirming the hypothesis previously introduced in Section 9.1 (Equation 9.1), in which at low SNR, the $K_{\text{Quiroga}}/\text{SNR}$ ratio is comparatively large, so moving from $K_{\text{Quiroga}} = 3$ to $K_{\text{Quiroga}} = 4$ produces a proportionally larger shift in threshold position relative to the spike peak, which translates into a measurable difference in crossing time.

It is worth clarifying that this SNR dependence does not stem from the comparator's finite output slew rate, which is a fixed device constant ($\approx 3.5 \text{ mV}/\mu\text{s}$), that determines the time required for its output to ramp up to the FPGA logic-high threshold, on the order of a few hundred microseconds ($V_{\text{IH}}/\text{SR} \approx 570 \mu\text{s}$ for $V_{\text{IH}} \approx 2 \text{ V}$). This value is identical in every condition and cannot, by itself, account for the monotonic, several-fold variation of τ_{system} with SNR. On the contrary, the longest latencies occur precisely at the lowest SNR, where the small overdrive near the elevated threshold keeps the amplifier in a bandwidth-limited, pseudo-linear regime rather than a slew-limited one. The finite slew rate therefore fixes the absolute order of magnitude of τ_{system} , whereas its variation across conditions is governed by the $K_{\text{Quiroga}}/\text{SNR}$ ratio through the position of the threshold on the rising edge.

Taken together, these results show that $K_{\text{Quiroga}} = 3$ is preferable not only in terms of detection accuracy, but also in terms of system latency, across the SNR range where the two studies overlap (SNR = 8–10). This is consistent with the recommendation of Section 9.2, where $K_{\text{Quiroga}} = 3$ was identified as the optimal value for the majority of practical operating conditions, based on the F_1 -score analysis of Study 1 across the SNR– f_r space.

From a practical standpoint, the range of τ_{system} values reported in Section 8.4.2 (Tables 8.10 and 8.11), from approximately $270 \mu\text{s}$ at SNR = 20 up to 1.5 ms at SNR = 8, $K_{\text{Quiroga}} = 4$, is relevant to real-time closed-loop stimulation applications. Two independent FPGA-based systems reported in the literature provide a useful point of reference, although neither allows for a strict, like-for-like benchmark.

Müller et al. [46] detect spikes through a threshold crossing on a band-pass filtered signal, conceptually the same mechanism employed here, and report the detection-to-stimulus-trigger delay relative to the spike's negative peak: $200\text{--}400 \mu\text{s}$, with jitter below $50 \mu\text{s}$. In terms of magnitude, their reported delay falls within the lower portion of the τ_{system} range measured here, closest to the high-SNR condition (SNR = 20: $269\text{--}435 \mu\text{s}$); their system operates on a fully digital, higher-sampling-rate signal path (20 kHz, on-chip filtering),

whereas τ_{system} additionally includes the analog front-end of the present platform, which is a plausible source of at least part of the residual difference.

Murphy et al. [47] adopt a different detection strategy, a multi-sample state machine designed to reject movement- and chewing-related artefacts, and report an analogous detection latency, again referenced to the spike peak, of 300–800 μs , to which a further 167 μs of downstream computational latency is added before stimulus delivery. They separately specify two closed-loop requirements for reliable activity-dependent stimulation: jitter below 1 ms and total latency below 10 ms. Since τ_{system} is a detection-only quantity and any subsequent actuation stage can only add further delay, it constitutes a lower bound on the round-trip latency the present platform could achieve once a stimulation stage is integrated; against Murphy et al.’s 10 ms criterion, this lower bound leaves a margin of at least 8.5 ms even in the worst tested condition.

9.6 Limitations

The experimental validation presented in this thesis is subject to several methodological limitations that should be considered when interpreting the results.

Spike waveforms were generated using a fixed biphasic template and injected via a calibrated DAQ device, bypassing the analog front-end entirely. While this approach guarantees precise ground-truth knowledge and full reproducibility, it does not capture the morphological variability intrinsic to real extracellular recordings as they typically occur in a live neuronal culture.

A further limitation concerns the sampling of the $K_{\text{Quiroga}}/\text{SNR}$ transition zone. As noted in Section 9.1, the rapid performance degradation observed between $K_{\text{Quiroga}}/\text{SNR} = 0.5$ and 0.7 is supported by only two parameter combinations within Study 1. The pattern is internally consistent and physically motivated, but the precise location and steepness of the transition boundary cannot be established with confidence from the available data.

The statistical analysis of Study 1 is based on five independent sessions per condition, a sample size motivated by the scale of the experimental design (630 acquisitions in total) but limiting the statistical power of the Friedman test. With $n = 5$ blocks, the test may fail to reach significance in conditions where a real effect exists but is modest in magnitude. Kendall’s W partially compensates for this limitation by providing a measure of effect size that is independent of statistical power, but the risk of false negatives in the significance map (Table 8.5) cannot be excluded.

A related limitation concerns the Schmitt trigger comparator used on the motherboard, a TLV8544 (Section 3.2). This device is a nanopower operational amplifier, configured as a window comparator, selected primarily for its low power consumption, a design priority dictated by the need to scale to 60 simultaneous channels. Its datasheet explicitly cautions that, unlike a dedicated comparator, a nanopower op amp used in this configuration exhibits a markedly slow slew rate and a comparatively long propagation delay arising from internal frequency compensation, restricting its suitability to low-frequency input signals. This is consistent with the analog front-end being the dominant contributor to τ_{system} (Section 9.5), and suggests that a comparator with a faster response would likely reduce the end-to-end latency without requiring any change to the digital pipeline.

A further limitation specific to the $\text{SNR} = 20$ condition of Study 4 concerns a hardware

inconsistency identified during data collection. The Schmitt trigger output is the logical OR of two window-comparator branches, one comparing the direct signal against the threshold and the other comparing the inverted signal, so that both positive and negative deflections can assert the output. Because the calibrated threshold at $\text{SNR} = 20$ was low relative to the noise floor, the output rested at logic-high between spikes instead of the intended logic-low: then, as a spike occurred, the inverted-signal branch crossed below this low threshold as the lobe approached its peak, pulling the output low and returning high shortly afterwards, as the signal receded from the peak and the inverted branch rose back above threshold. The edge detector, designed to register a $0 \rightarrow 1$ transition at spike onset, therefore captured this recovery transition instead of the intended leading-edge crossing.

The scope of Study 4 is itself limited in ways that mirror the broader experimental design. Latency was characterized only under a single, fixed firing rate with a fresh calibration performed at each tested SNR condition; unlike Study 2, no protocol was run in which calibration is fixed at a reference SNR while the effective signal quality is subsequently allowed to drift within the same session, so it is not known whether τ_{system} behaves differently under a mismatched calibration than under the matched, per-condition calibration used here. Similarly, unlike Study 3, latency was not evaluated under Poisson-distributed spike timing; whether burst-like, irregular firing introduces additional latency variability, for instance through overlapping detection events or refractory-period interactions, was not investigated.

The comparison with literature values in Section 9.5 should be read with some caution: the two reference systems operate on biological signals with their own noise and amplitude statistics, use detection algorithms of different complexity, and were characterized under different experimental protocols, so the numbers offered there serve as context for the order of magnitude involved rather than as a controlled benchmark. Within these limits, τ_{system} appears broadly consistent with what has been reported for other FPGA-based, threshold-crossing detection systems, and comfortably satisfies the more conservative of the two published latency criteria; whether it also satisfies the stricter sub-millisecond figures depends on the operating SNR.

Chapter 10

Conclusions

This thesis designed and implemented a new digital backend based on the Lattice MachXO2-7000HE FPGA for an event-driven in vitro neural acquisition platform, replacing the previous ARM Cortex-M7 microcontroller and extending the number of simultaneous channels from 15 to 60 and confirming the low output data rate, reducing it by more than 99% relative to continuous acquisition at 10 kS/s across 60 channels.

The FPGA implementation, written entirely in SystemVerilog, applies in parallel the calibration pipeline, 60-channel event detection, and UART streaming to the host, while the MATLAB interface manages the entire acquisition pipeline, integrating the Quiroga-based hybrid algorithm for threshold estimation and providing real-time visualisation of neural activity.

The experimental validation, conducted using synthetic signals, revealed a clear correlation between calibration effectiveness and the resulting detection performance, demonstrating an explicit dependence on the ratio between the signal-to-noise ratio and the value of K_{Quiroga} employed. The studies also identified two distinct failure regimes: one limited by SNR and one by firing rate, the latter mediated by spike contamination of the calibration window.

In particular, an optimal calibration at the start of the acquisition session proved to be a fundamental requirement for effective neural spike detection, as demonstrated in Study 2 when, even in the presence of signal morphology degradation within the same session with a consequent degradation of SNR, the device maintains high performance and is capable of correctly detecting spike rising edges, favoring precision over recall. This behavior makes the system more conservative, reducing the probability of misclassifying noise events as real spikes, at the cost of a higher rate of false negatives under the most demanding operating conditions. Moreover, it has been demonstrated that Poisson spike timing, being closer to realistic experimental conditions, produces a more gradual performance degradation than deterministic timing at the same mean firing rate.

Beyond detection performance, the temporal behaviour of the platform was characterized in Study 4, which separated the overall acquisition latency into a digital and an analog contribution. The FPGA-internal latency matched the theoretical prediction of the digital pipeline almost exactly, confirming the correctness of the RTL implementation and showing that the digital backend contributes a negligible fraction of the overall system response. The end-to-end latency was instead found to be governed almost entirely by the

analog front-end, and to depend on the same $K_{\text{Quiroga}}/\text{SNR}$ ratio that governs detection accuracy: $K_{\text{Quiroga}} = 3$ emerged as preferable not only in terms of accuracy, but also in terms of response time, reinforcing it as the recommended operating point across the range of conditions examined. These latency figures were found to be broadly consistent, in order of magnitude, with values reported for comparable FPGA-based, threshold-crossing acquisition systems in the literature.

Future developments will prioritise validation on real neuronal cultures, where spike amplitudes vary across neurons and electrodes, waveform morphology depends on the neuron-electrode distance and orientation, and background noise exhibits non-stationary properties driven by network bursting activity. Such conditions cannot be fully reproduced using synthetic signals and may affect both the calibration algorithm and detection performance in ways not fully captured by the present study. On the hardware side, the low-power Schmitt trigger comparator could be substituted with a faster device to reduce end-to-end latency further, and the platform could be extended with an integrated stimulation stage to enable genuine closed-loop operation, building on the latency margin identified in this work.

Bibliography

- [1] B. Kagan *et al.*, “In vitro neurons learn and exhibit sentience when embodied in a simulated game-world,” *Neuron*, vol. 110, pp. 3952–3969.e8, 2022. Available: [https://www.cell.com/neuron/fulltext/S0896-6273\(22\)00806-6](https://www.cell.com/neuron/fulltext/S0896-6273(22)00806-6).
- [2] M. Serra *et al.*, “A low-cost interface for multi-electrode array data acquisition systems,” *BioTechniques*, vol. 45, pp. 451–456, 2008. Available: <https://www.tandfonline.com/doi/full/10.2144/000112911>.
- [3] A. Mazzoni, F. D. Broccard, E. Garcia-Perez, P. Bonifazi, M. E. Ruaro, and V. Torre, “On the dynamics of the spontaneous activity in neuronal networks,” *PLOS ONE*, vol. 2, no. 5, p. e439, 2007.
- [4] D. A. Wagenaar, R. Madhavan, J. Pine, and S. M. Potter, “Controlling bursting in cortical cultures with closed-loop multi-electrode stimulation,” *Journal of Neuroscience*, vol. 25, no. 3, pp. 680–688, 2005.
- [5] M. S. Tanveer, D. Patel, H. E. Schweiger, K. D. Abu-Bonsrah, B. Watmuff, A. Azadi, S. Pryshchep, K. Narayanan, C. Puleo, K. Natarajan, M. A. Mostajo-Radji, B. J. Kagan, and G. Wang, “Starting a synthetic biological intelligence lab from scratch,” *Patterns*, vol. 6, no. 5, p. 101232, 2025.
- [6] D. E. Goldman, “Potential, impedance, and rectification in membranes,” *Journal of General Physiology*, vol. 27, no. 1, pp. 37–60, 1943.
- [7] A. L. Hodgkin and B. Katz, “The effect of sodium ions on the electrical activity of the giant axon of the squid,” *Journal of Physiology*, vol. 108, no. 1, pp. 37–77, 1949.
- [8] D. A. McCormick, “Chapter 5 - membrane potential and action potential,” in *Fundamental Neuroscience* (L. R. Squire, D. Berg, F. E. Bloom, S. du Lac, A. Ghosh, and N. C. Spitzer, eds.), pp. 93–116, Academic Press, 4th ed., 2013. Available: <https://doi.org/10.1016/B978-0-12-385870-2.00005-6>.
- [9] A. L. Hodgkin and A. F. Huxley, “A quantitative description of membrane current and its application to conduction and excitation in nerve,” *Journal of Physiology*, vol. 117, no. 4, pp. 500–544, 1952.
- [10] K. H. Pettersen, E. Hagen, and G. T. Einevoll, “Extracellular spikes and CSD,” in *Handbook of Neural Activity Measurement* (R. Brette and A. Destexhe, eds.), ch. 4, pp. 92–135, Cambridge University Press, 2012.
- [11] Y. J. Jung, S. H. Sun, A. Almasi, M. Yunzab, H. Meffin, and M. R. Ibbotson, “Characterization of extracellular spike waveforms recorded in wallaby primary visual cortex,” *Frontiers in Neuroscience*, vol. 17, p. 1244952, 2023.
- [12] M. E. Spira and A. Hai, “Multi-electrode array technologies for neuroscience and cardiology,” *Nature Nanotechnology*, vol. 8, no. 2, pp. 83–94, 2013. Available: <https://doi.org/10.1038/nnano.2012.214>.

- [//www.nature.com/articles/nnano.2012.265](https://www.nature.com/articles/nnano.2012.265).
- [13] H. G. Rey, C. Pedreira, and R. Quiñan Quiroga, “Past, present and future of spike sorting techniques,” *Brain Research Bulletin*, vol. 119, pp. 106–117, 2015.
 - [14] M. E. J. Obien *et al.*, “Revealing neuronal function through microelectrode array recordings,” *Frontiers in Neuroscience*, vol. 8, p. 423, 2015. Available: <https://www.frontiersin.org/articles/10.3389/fnins.2014.00423/full>.
 - [15] C. Gold, D. A. Henze, C. Koch, and G. Buzsáki, “On the origin of the extracellular action potential waveform: a modeling study,” *Journal of Neurophysiology*, vol. 95, no. 5, pp. 3113–3128, 2006.
 - [16] C. Hammond, “Chapter 6: The chemical synapses,” in *Cellular and Molecular Neurophysiology*, pp. 121–144, Boston: Academic Press (Elsevier), 4th ed., 2015. Available: <https://doi.org/10.1016/B978-0-12-397032-9.00006-6>.
 - [17] B. L. Roth, “Molecular pharmacology of metabotropic receptors targeted by neuropsychiatric drugs,” *Nature Structural & Molecular Biology*, vol. 26, pp. 535–544, 2019. Available: <https://doi.org/10.1038/s41594-019-0252-8>.
 - [18] D. Purves *et al.*, “Excitatory and inhibitory postsynaptic potentials,” in *Neuroscience*, Sunderland, MA: Sinauer Associates, 2nd ed., 2001. Available: <https://www.ncbi.nlm.nih.gov/books/NBK11117/>.
 - [19] A. Pereda, “Electrical synapses and their functional interactions with chemical synapses,” *Nature Reviews Neuroscience*, vol. 15, pp. 250–263, 2014. Available: <https://doi.org/10.1038/nrn3708>.
 - [20] C.-P. Segeritz and L. Vallier, “Cell culture: Growing cells as model systems in vitro,” in *Basic Science Methods for Clinical Researchers*, pp. 151–172, Boston: Academic Press, 2017. Available: <https://doi.org/10.1016/B978-0-12-803077-6.00009-6>.
 - [21] M.-T. Arango *et al.*, “Cell culture and cell analysis,” in *Autoimmunity: From Bench to Bedside*, ch. 45, Bogota: El Rosario University Press, 2013.
 - [22] A. Verma, M. Verma, and A. Singh, “Animal tissue culture principles and applications,” in *Animal Biotechnology*, pp. 269–293, Cambridge, MA: Academic Press, 2020. Available: <https://doi.org/10.1016/B978-0-12-811710-1.00012-4>.
 - [23] C. Suñol *et al.*, “Studies with neuronal cells: From basic studies of mechanisms of neurotoxicity to the prediction of chemical toxicity,” *Toxicology in Vitro*, vol. 22, no. 5, pp. 1350–1355, 2008. Available: <https://doi.org/10.1016/j.tiv.2008.03.009>.
 - [24] Leica Microsystems, “Introduction to mammalian cell culture.” Available: <https://www.leica-microsystems.com/science-lab/life-science/introduction-to-mammalian-cell-culture/>. [Accessed: 11-Apr-2026].
 - [25] E. Neher and B. Sakmann, “The patch clamp technique,” *Scientific American*, vol. 266, no. 3, pp. 44–51, 1992. Available: <https://www.jstor.org/stable/24938980>.
 - [26] Y. Nam and B. C. Wheeler, “In vitro microelectrode array technology and neural recordings,” *Critical Reviews in Biomedical Engineering*, vol. 39, no. 1, pp. 45–61, 2011. Available: <https://www.dl.begellhouse.com/journals/4b27cbfc562e21b8,7d5e987c7e576915,41cf5d45336340f0.html>.
 - [27] N. Shmoel, N. Rabieh, S. M. Ojovan, H. Erez, E. Maydan, and M. E. Spira, “Multisite electrophysiological recordings by self-assembled loose-patch-like junctions between cultured hippocampal neurons and mushroom-shaped microelectrodes,” *Scientific Reports*, vol. 6, p. 27110, 2016.

-
- [28] Multi Channel Systems MCS GmbH, “Microelectrode arrays.” Available: <https://www.multichannelsystems.com/products/microelectrode-arrays>. [Accessed: 11-Apr-2026].
 - [29] D. R. Merrill, M. Bikson, and J. G. R. Jefferys, “Electrical stimulation of excitable tissue: design of efficacious and safe protocols,” *Journal of Neuroscience Methods*, vol. 141, no. 2, pp. 171–198, 2005. Available: <https://doi.org/10.1016/j.jneumeth.2004.10.020>.
 - [30] S. F. Lempka, M. D. Johnson, D. W. Barnett, M. A. Moffitt, K. J. Otto, D. R. Kipke, and C. C. McIntyre, “Optimization of microelectrode design for cortical recording based on thermal noise considerations,” in *Proceedings of the 28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)*, pp. 3361–3364, 2006.
 - [31] R. R. Harrison and C. Charles, “A low-power low-noise cmos amplifier for neural recording applications,” *IEEE Journal of Solid-State Circuits*, vol. 38, no. 6, pp. 958–965, 2003. Available: <https://doi.org/10.1109/JSSC.2003.811979>.
 - [32] R. Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul, “Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering,” *Neural Computation*, vol. 16, no. 8, pp. 1661–1687, 2004. Available: <https://ieeexplore.ieee.org/abstract/document/6790337>.
 - [33] Multi Channel Systems MCS GmbH, “Mea2100-system.” <https://www.multichannelsystems.com/products/mea2100-systems>. Accessed: 2025.
 - [34] J. Müller *et al.*, “High-resolution cmos mea platform to study neurons at subcellular, cellular, and network levels,” *Lab on a Chip*, vol. 15, no. 13, pp. 2767–2780, 2015.
 - [35] Axion BioSystems, “Maestro pro mea system.” <https://www.axionbiosystems.com/products/mea/maestro-pro>. Accessed: 2025.
 - [36] Intan Technologies, “Rhd recording system.” https://intantech.com/RHD_system.html. Accessed: 2025.
 - [37] Intan Technologies, “Rhd rhythm interface.” https://intantech.com/products_RHD2000_Rhythm.html. Accessed: 2025.
 - [38] L. D. Garma, L. Matino, G. Melle, F. Moia, F. De Angelis, F. Santoro, and M. Dipalo, “Cost-effective and multifunctional acquisition system for in vitro electrophysiological investigations with multi-electrode arrays,” *PLOS ONE*, vol. 14, no. 3, p. e0214017, 2019.
 - [39] ARM Limited, *Cortex-M7 Processor Technical Reference Manual*. ARM Limited, 2014. Revision r1p1, ARM DDI 0489D. Available: <https://developer.arm.com/documentation/ddi0489/latest>.
 - [40] U. Farooq, Z. Marrakchi, and H. Mehrez, “Fpga architectures: An overview,” in *Tree-Based Heterogeneous FPGA Architectures*, pp. 7–29, New York: Springer, 2012.
 - [41] M. Hassan, F. Awwad, M. Atef, and O. Hasan, “Approximate computing-based processing of MEA signals on FPGA,” *Electronics*, vol. 12, no. 4, p. 848, 2023.
 - [42] M. Boscherini, A. Pjeci, G. Tomagra, F. Picollo, and A. Sanginario, “A low-cost and ultra-low-data-rate event-driven platform for in-vitro real-time multichannel neural signal acquisition,” in *Proceedings of the IEEE Mediterranean Electrotechnical Conference (MELECON)*, IEEE, 2026.
 - [43] R. H. Barker, “Group synchronizing of binary digital systems,” in *Communication Theory* (W. Jackson, ed.), pp. 273–287, London, UK: Butterworth, 1953.

- [44] D. N. Hill, S. B. Mehta, and D. Kleinfeld, “Quality metrics to accompany spike sorting of extracellular signals,” *Journal of Neuroscience*, vol. 31, no. 24, pp. 8699–8705, 2011.
- [45] M. Tambaro, M. Bisio, M. Maschietto, A. Leparulo, and S. Vassanelli, “FPGA design integration of a 32-microelectrodes low-latency spike detector in a commercial system for intracortical recordings,” *Digital*, vol. 1, no. 1, pp. 34–53, 2021.
- [46] J. Müller, D. J. Bakkum, and A. Hierlemann, “Sub-millisecond closed-loop feedback stimulation between arbitrary sets of individual neurons,” *Frontiers in Neural Circuits*, vol. 6, p. 121, 2013.
- [47] M. Murphy, S. Buccelli, Y. Bornat, D. Bundy, R. Nudo, D. Guggenmos, and M. Chiappalone, “Improving an open-source commercial system to reliably perform activity-dependent stimulation,” *Journal of Neural Engineering*, vol. 16, no. 6, p. 066022, 2019.
- [48] M. Friedman, “The use of ranks to avoid the assumption of normality implicit in the analysis of variance,” *Journal of the American Statistical Association*, vol. 32, no. 200, pp. 675–701, 1937.
- [49] Y. Benjamini and Y. Hochberg, “Controlling the false discovery rate: a practical and powerful approach to multiple testing,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 57, no. 1, pp. 289–300, 1995.
- [50] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*. Hillsdale, NJ: Lawrence Erlbaum Associates, 2nd ed., 1988.
- [51] F. Wilcoxon, “Individual comparisons by ranking methods,” *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.