UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
CURSO DE ENGENHARIA ELETRÔNICA

Leonardo Baltazar Clivati

Design and Analysis of a Digital Lock-in Amplifier implemented in a FPGA

Florianópolis
2025

Leonardo Baltazar Clivati

Design and Analysis of a Digital Lock-in Amplifier implemented in a FPGA

Trabalho de Conclusão de Curso do Curso de Engenharia Eletrônica do Centro Tecnológico da Universidade Federal de Santa Catarina para a obtenção do título de Bacharel em Engenharia Eletrônica.
Supervisor:: Prof. Fernando Rangel de Sousa, Dr.

Florianópolis
2025

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Clivati, Leonardo Baltazar
Design and Analysis of a Digital Lock-in Amplifier implemented in a FPGA / Leonardo Baltazar Clivati ; orientador, Fernando Rangel de Sousa, 2025.
54 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia Eletrônica, Florianópolis, 2025.

Inclui referências.

1. Engenharia Eletrônica. 2. Amplificador Lock-In. 3. Circuitos Digitais. 4. Aritmética de Ponto Fixo. I. de Sousa, Fernando Rangel . II. Universidade Federal de Santa Catarina. Graduação em Engenharia Eletrônica. III. Título.

Leonardo Baltazar Clivati

Design and Analysis of a Digital Lock-in Amplifier implemented in a FPGA

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "Bacharel em Engenharia Eletrônica" e aprovado em sua forma final pelo Curso de Engenharia Eletrônica.

Florianópolis, 16 de julho de 2025.

Documento assinado digitalmente
Daniela Ota Hisayasu Suzuki
Data: 17/07/2025 17:42:45-0300
CPF: ***.397.389-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Daniela Ota Hisayasu Suzuki, Dra.
Coordenadora do Curso

Banca Examinadora:

Documento assinado digitalmente
Fernando Rangel de Sousa
Data: 18/07/2025 14:56:57-0300
CPF: ***.649.114-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Fernando Rangel de Sousa, Dr.
Supervisor
EEL-UFSC

Documento assinado digitalmente
Daniel Juan Pagano
Data: 16/07/2025 13:05:30-0300
CPF: ***.839.419-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Daniel Juan Pagano, Dr.
Avaliador
DAS-UFSC

Documento assinado digitalmente
Jefferson Luiz Brum Marques
Data: 16/07/2025 12:15:35-0300
CPF: ***.624.060-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Jefferson Luiz Brum Marques, Dr.
Avaliador
EEL-UFSC

ACKNOWLEDGEMENTS

Sem o apoio dos meus pais, Keila e Geraldo, eu não teria chegado onde estou hoje. Independentemente do que eu precisava, eles sempre estiveram ao meu lado. Quando necessário, apontavam meus erros com carinho e procuravam me explicar seu ponto de vista. Em alguns momentos, podia parecer que eu não dava atenção, mas sempre refletia sobre cada conselho recebido — e tenho certeza de que todos foram fundamentais para a minha formação.

Agradeço também aos meus avós, Raul e Solange, por sempre acreditarem no meu potencial e por contribuírem significativamente na minha formação como pessoa. Eles desempenharam um papel importante na minha educação básica, sempre enfatizando o valor do respeito ao próximo.

Minha irmã, Beatriz, também teve um papel essencial na minha motivação. Sempre quis dar o melhor exemplo possível para ela. Sua constante evolução me inspira a seguir firme na minha jornada de autodesenvolvimento.

Minha tia, Fernanda, além de oferecer apoio emocional, sempre me aconselhou sobre como enfrentar os desafios da universidade e aproveitar ao máximo essa experiência. Sem seus conselhos, tenho certeza de que não teria explorado tantas oportunidades acadêmicas e profissionais ao longo da graduação.

Ao professor Fernando, agradeço pela paciência e pela orientação durante o desenvolvimento deste trabalho. Aos professores Daniel, Francisco e Adriano, sou grato pelo suporte técnico e pela disponibilização dos componentes e equipamentos necessários.

ABSTRACT

This work presents the design and implementation of a digital Lock-In Amplifier (LIA) system based on FPGA technology, motivated by the need to monitor cancer cell proliferation via impedance tomography following drug administration. The application demands the recovery of weak signals in highly noisy environments—a challenge well-suited to the lock-in amplification technique. The study begins by outlining the principles of lock-in detection and its adaptation to digital platforms using FPGAs. Key theoretical foundations are explored, including impedance measurement methods and the design of digital filters such as FIR, IIR, and decimation filters, along with their architectural implications. The system architecture was developed through a two-phase approach: initial algorithm validation using Python with floating-point precision, followed by a resource-efficient fixed-point implementation. Detailed numerical analysis guided the precision trade-offs. RTL synthesis results are presented to validate the higher-level implementations and analyze the FPGA resource usage. The developed LIA system is evaluated in terms of input noise, SNR-dependent system error, dynamic reserve, sensitivity and vertical resolution. Although, the average magnitude error was significand as the SNR reduced, a standard deviation of 3mV in magnitude and 80mrad in phase was achieved at an SNR of –36dB. These results suggest that, with proper calibration, the error can be reduced to just a few millivolts and a few milliradians. The conclusion contain an analysis of the design and suggestions of improvement.

Keywords: Digital Lock-in Amplifier, FPGA, LIA, Fixed-Point Arithmetic.

RESUMO

Este trabalho apresenta o projeto e a implementação de um Amplificador Lock-In (LIA) digital implementado em uma FPGA, motivado pela necessidade de monitorar a proliferação de células cancerígenas por meio de tomografia de impedância após a administração de fármacos. A aplicação exige a recuperação de sinais fracos em ambientes altamente ruidosos — um desafio para o qual a técnica de amplificação lock-in é particularmente adequada. O estudo começa descrevendo os princípios da detecção lock-in e sua adaptação a plataformas digitais utilizando FPGAs. São explorados os fundamentos teóricos principais, incluindo métodos de medição de impedância e o projeto de filtros digitais como FIR, IIR e filtros de decimação. A arquitetura do sistema foi desenvolvida por meio de uma abordagem em duas fases: validação inicial dos algoritmos usando Python com precisão em ponto flutuante, seguida por uma implementação em ponto fixo otimizada para uso eficiente de recursos. Uma análise numérica detalhada guiou as decisões de precisão. Os resultados da síntese RTL são apresentados para validar as implementações de alto nível e analisar o uso de recursos da FPGA. O LIA desenvolvido foi avaliado em termos de ruído de entrada, erro do sistema dependente da SNR, reserva dinâmica, sensibilidade e resolução vertical. Embora o erro médio de magnitude tenha sido significativo com a redução da SNR, uma média de desvio padrão de 3 mV na magnitude e 80 mrad na fase foi alcançada com uma SNR de –36 dB. Esses resultados sugerem que, com a calibração adequada, o erro pode ser reduzido a apenas alguns milivolts e poucos miliradianos. A conclusão contêm uma análise do circuito e sugetões de aprimoramento.

Keywords: Amplificador Lock-In Digital, FPGA, LIA, Aritmética de Ponto Fixo.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

ADC	Analog-to-Digital Converter
ALM	Adaptative Logic Modules
CDF	Cumulative Density Function
CIC	Cascaded Integrator-Comb
CORDIC	Coordinate Rotation Digital Computer
DAC	Digital-to-Analog Converter
DC	Direct Current
DDS	Direct Digital Synthesizer
DUT	Device Under Test
ENOB	Effective Number of Bits
FIR	Finite Impulse Response
FP	Fixed-Point
FPGA	Field-Programmable Gate Arrays
HPS	Hard Processor System
IIR	Infinite Impulse Response
IP	Intellectual Property
LIA	Lock-In Amplifier
LUT	Look-Up Tables
NCO	Numerically Controlled Oscillator
RBC	Red Blood Cell
RTL	Register-Transfer Level
SINAD	Signal-to-Noise and Distortion Ratio
SNR	Signal-to-Noise Ratio

LIST OF SYMBOLS

M	Prefix Mega (10^6)
Hz	Hertz
k	Prefix kilo (10^3)
sps	Samples per second
n	Prefix nano (10^{-9})
V	Volt
ppm	Parts per million
dB	Decibels
μ	Prefix micro (10^{-6})
m	Prefix mili (10^{-3})
rad	Radians
Ω	Ohm
F	Farad
A	Ampere

CONTENTS

1 INTRODUCTION

1.1 MOTIVATION

Many different applications require the extraction of small signals from high-noise environments, such as radio astronomy, nuclear magnetic resonance, and spectrometry (AGUIAR SANTOS et al., 2016). The lock-in technique is a well-established method for extracting such signals, leveraging phase-sensitive detection to recover weak signals buried in significant noise (KISHORE; AKBAR, 2020). Lock-In Amplifier (LIA) can be implemented using either analog or digital approaches. The digital implementation offers several advantages; for example, the multiplication stage in analog designs typically incurs higher costs and requires more hardware for precision and quality. In contrast, digital LIAs utilize discrete multipliers, which are generally cheaper and offer precision directly related to their digital resolution (MACIAS-BOBADILLA et al., 2016).

Furthermore, digital systems can be efficiently implemented on Field-Programmable Gate Arrays (FPGA), which are integrated circuits that allow complete hardware reconfiguration according to the specific application (AGUIAR SANTOS et al., 2016; MACIAS-BOBADILLA et al., 2016). FPGAs are particularly appealing for LIA implementations due to their ability to handle high-speed data acquisition and enable the design of filters with variable time constants. They provide a reconfigurable architecture and parallel operation, crucial for applications requiring control, generation, acquisition, and signal processing of high-frequency signals in real-time. Modern FPGA development boards further simplify the implementation of digital LIAs by integrating essential components such as Analog-to-Digital Converter (ADC), memory, and real-time processing capabilities (AGUIAR SANTOS et al., 2016). This allows for the development of programmable and flexible devices, capable of performing fast real-time execution at high-frequency modulations, extending up to 10 MHz, and even up to 600 MHz in commercial designs (KISHORE; AKBAR, 2020). The use of FPGAs also facilitates the generation and selection of appropriate frequencies for various experiments, offering good frequency stability (MACIAS-BOBADILLA et al., 2016).

The initial motivation for this project emerged from discussions with a laboratory at the university hospital, which required a solution to monitor the proliferation of cancer cells in response to various drug treatments. The proposed method involved using impedance tomography, where a known low current is applied to the cell culture, the resulting voltage is measured, and the impedance is calculated. Due to the low amplitude of the signal and its high susceptibility to noise, a LIA circuit is an appropriate choice for accurately recovering and monitoring the signal in this context. Researchers currently monitor cells using a microscope, a time-consuming process prone to human error. Implementing a specialized circuit can enhance data quality and enable the development of studies to model cell multiplication behavior more effectively.

This work proposes the development of a LIA implemented on an FPGA platform, with an emphasis on understanding and validating the design process of a digital LIA. Commercial LIA systems are often expensive, limiting their use in experimental and academic environments, therefore this work also lay the groundwork for the developing of a more cost-effective implementation in the future. The project was developed using a DE10-Nano FPGA board and the Quartus design environment, with a focus on evaluating different low-pass filter architectures and proposing optimizations for the internal digital circuits. The characterization process and comparison with a commercial LIA revealed areas where the design needs improvement. Additionally, the metrics obtained can serve as benchmarks for future projects, helping to track the progress and evolution of the design.

1.2 DOCUMENT STRUCTURE

This thesis is organized into five main chapters. Chapter 2 provides the theoretical foundation required for implementing a Lock-In Amplifier, covering the lock-in detection algorithm, digital filtering techniques, and an analysis of errors in digital systems. Chapter 3 details digital LIA implementations for different applications and highlights the performance of each circuit. Chapter 4 presents the initial implementation results obtained using Python, followed by the FPGA emulation outcomes. It also includes a comparison of circuit designs and a discussion of the design decisions made for the final LIA implementation. Chapter 5 details the circuit characterization process and compares the results with those of a commercial LIA. Finally, chapter 6 summarizes the main findings and offers recommendations for future work.

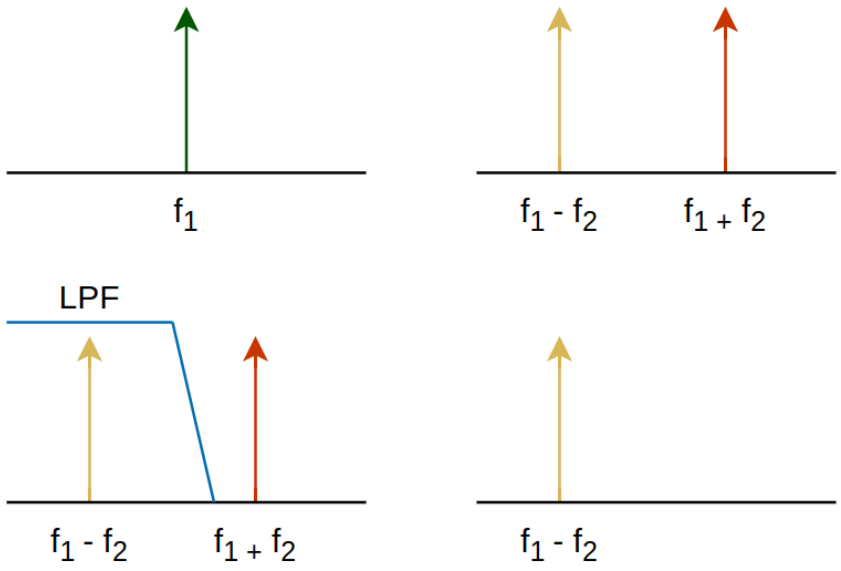
2 THEORETICAL BACKGROUND

This section introduces the key concepts underlying the LIA design, with an overview of the lock-in technique. The discussion then extends to impedance measurement principles, followed by an in-depth exploration of digital filters, including Finite Impulse Response (FIR), Infinite Impulse Response (IIR), and decimation filters, along with their architectural considerations. The section concludes with a review of fixed-point and floating-point number representations, and methods for analyzing numerical errors in digital systems.

2.1 LOCK-IN AMPLIFIER: AN INTRODUCTION

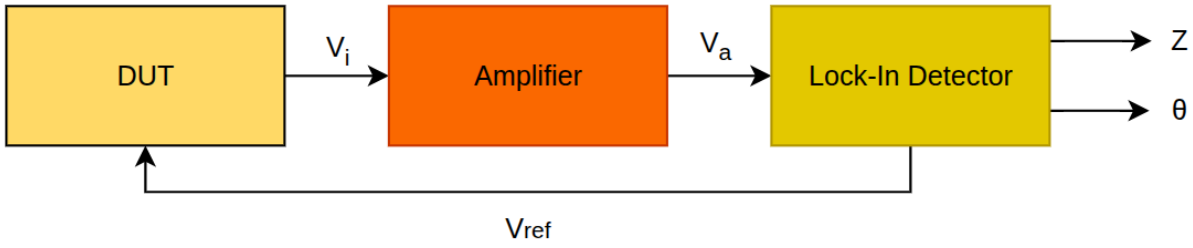
The basic principle behind the LIA is illustrated in Figure 1, which explores operations in the frequency domain to extract signals in high-noise environments. The input signal $V(t)$ is multiplied by the reference signal $R(t)$ to dislocate the desired frequency component and filter the noise. Consider that $V_a(t) = A\ sin(2\pi f_1 t + \theta_1) + n(t)$, where $n(t)$ is the noise component and $R(t) = B\ sin(2\pi f_2 t + \theta_2)$ is the reference signal. When the signals are multiplied by the reference signal, they are dislocated in the spectrum, generating the components in Equation 1.

Figure 1 – LIA Operations in the Frequency Domain



$$\begin{aligned} V_a(t) \times R(t) &= [A\ sin(2\pi f_1 t + \theta_1) + n(t)] \times B\ sin(2\pi f_2 t + \theta_2) \\ &= AB\ sin(2\pi f_1 t + \theta_1) \times sin(2\pi f_2 t + \theta_2) + R(t) \times n(t) \\ &= \frac{AB}{2}[cos(2\pi(f_1 - f_2)t + \theta_1 - \theta_2) - cos(2\pi(f_1 + f_2)t + \theta_1 + \theta_2)] + R(t) \times n(t) \end{aligned} \quad (1)$$

Figure 2 – Block Diagram Lock-In Amplifier



After the result from $V_a(t) \times R(t)$ passes through a low-pass filter, the noise component and $\frac{AB}{2}cos[2\pi(f_1+f_2)t+\theta_1+\theta_2]$ are removed, resulting in $\frac{AB}{2}cos[2\pi(f_1-f_2)t+\theta_1-\theta_2]$. If $f_1 - f_2 = 0$ the resulting signal is a Direct Current (DC) component. This is why the reference signal is chosen to have the same frequency as the desired component of the input signal—ensuring that the desired signal is shifted to DC for effective extraction. The phase difference $\theta_1 - \theta_2$ is a known value.

To further improve the result and eliminate the influence of the phase difference between the input and reference, the input is multiplied by the reference quadrature $R_q(t)$. Consequently, after filtering, the system has in-phase X and quadrature Y components, when $f_1 = f_2$:

$$X = \frac{AB}{2}cos(\theta_1 - \theta_2) \tag{2}$$

$$Y = \frac{AB}{2}sen(\theta_1 - \theta_2) \tag{3}$$

To calculate module Z and phase θ, the following equations are applied:

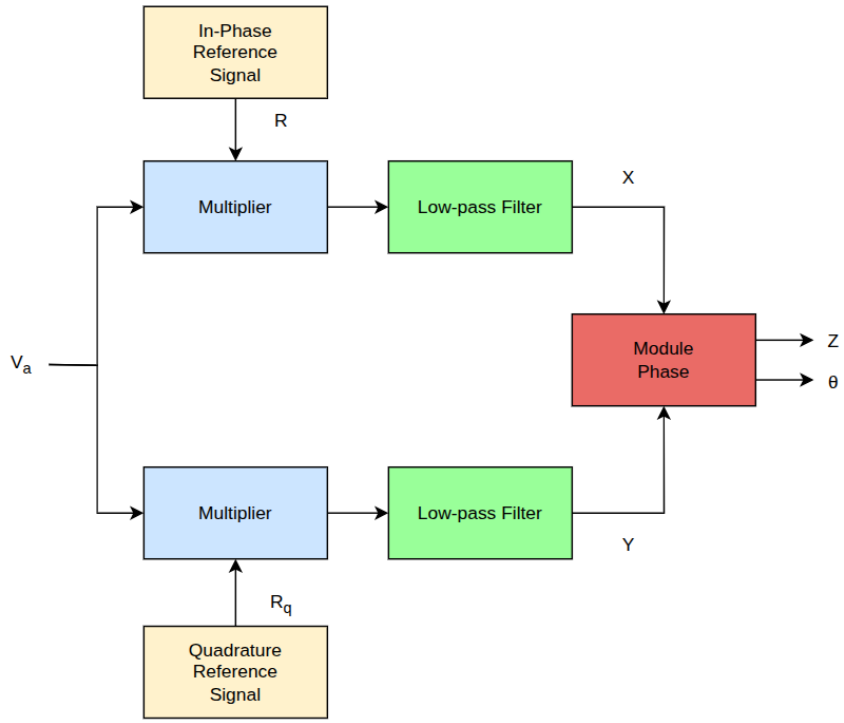
$$Z = A = \frac{2}{B}\sqrt{X^2 + Y^2} \tag{4}$$

$$\theta = arctan(Y/X) \tag{5}$$

The filter is the main component of the system because as the filter becomes more selective, the time constant increases; however, the noise effect in the output is lower. Although reducing the selectivity decreases the time constant, increasing the output sample rate, the noise has a greater influence on the result.

Due to the structure of the LIA algorithm, it is capable of recovering signals in very low SNR environments. As illustrated in Figure 1, after the filtering stage, only the component at frequency $f_1 - f_2$ remains—resulting in a DC signal when $f_1 = f_2$. With a properly designed low-pass filter, considering both its bandwidth and order, the surrounding noise can be effectively eliminated from the measurement. An additional advantage of this technique is the flexibility to select the modulation frequency. By choosing an appropriate frequency, it is possible to avoid spectral regions with high noise content, thereby improving the overall quality and reliability of the measurement(POTTS, 2023).

Figure 3 – Block Diagram Lock-In Detector



For example, when measuring impedance, interference from flicker noise (1/f) can be removed by selecting an appropriate reference frequency, thereby shifting the measurement to a specific region of the spectrum.

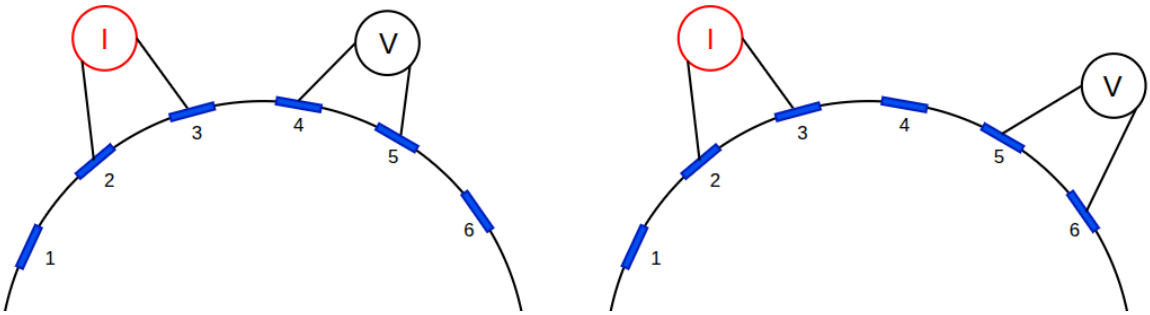
Figure 2 illustrates the components of the LIA, while Figure 3 provides a detailed view of the lock-in detector, which is the primary focus of this thesis. Given that the scope of this work centers on the digital circuitry, the term LIA is used throughout the document to refer specifically to the lock-in detector.

2.2 IMPEDANCE MEASUREMENT

The motivation for LIA design was to monitor cancer cell multiplication behavior, and impedance tomography is a good option for this purpose. Impedance has two components, resistive and reactive. To retrieve the impedance from the Device Under Test (DUT), a known current I is applied and the resulting voltage V is measured. Subsequently, according to Ohm's first law, as shown in Figure 4.

The output returns the voltage module and phase, and the impedance is obtained by dividing the voltage module by the current module. In impedance tomography, an association of electrodes is used to stimulate the DUT and recover the response, which is the distribution of electrodes in a circle; then, using multiplexes, the current is applied in different pairs of electrodes and measured at different points(AGUIAR SANTOS et al., 2016). From these measurements, it is possible to build an image, and the image resolution

Figure 4 – Electrode Arrangement for Impedance Tomography



Source: (AGUIAR SANTOS et al., 2016)

is proportional to the number of electrodes.

2.3 DIGITAL FILTERS

The filter is the main component of a digital LIA, and the quality and sample rate of the output depend directly on its architecture. It is important to analyze and compare different filter designs, considering the possible restrictions of the implementation step.

2.3.1 Finite Impulse Response

The finite impulse response filter has a transfer function $h[k]$ represented by a finite sequence of non-zero impulse responses, as illustrated in Equation 6. The order of the filter M is directly related to the transition bandwidth and attenuation in the stopband. As the order increases, the mentioned parameters decrease, and the filter exhibits a more selective response. The problem of increasing the order is the hardware necessary to implement new additions and store more coefficients.

$$y[n] = \sum_{i=0}^{M-1} h[i]x[n-i] \tag{6}$$

The FIR project can start with the ideal impulse response, defining the desired cutoff frequency, transition bandwidth, and stopband attenuation, and the next step is to truncate the impulse response. After this process, the frequency domain response still presents ripples; to fix this problem, the response is windowed (BOURKE, 1999). This type of filter presents stability, a linear phase response, low sensitivity to quantization errors, and a simple implementation(WOODS et al., 2010).

2.3.2 Infinite Impulse Response

As stated in the name, the infinite impulse response filter has a transfer function $h[k]$ composed of an infinite sequence of impulse responses and a feedback term, as detailed

in Equation 7. The order has the same effect in the transition band and stop band as observed in FIR, but in this case, the number of coefficients required is lower when compared with a FIR, that has a similar response in the frequency domain.

$$y[n] = \sum_{i=0}^{M-1} b[i]x[n-i] - \sum_{j=1}^{M-1} a[j]y[n-j] \tag{7}$$

The IIR filter project usually starts with the project of an analog filter, for instance, Butterworth, Chebyshev, and Elliptic. The filter is then converted to a digital representation in the z-plane, as given in Equation 8. The poles have to be inside the unit circle to ensure stability, and this filter is more sensitive to quantization errors owing to feedback(WOODS et al., 2010).

$$H[z] = \frac{\sum_{i=0}^{M-1} b_i x_{n-i}}{1 + \sum_{j=1}^{M-1} a_j y_{n-j}} \tag{8}$$

This type of filter can be implemented using different structures: the Direct Form I, Direct Form II, Cascade Form and Parallel Form. The direct forms implement the filter directly from the difference equation with an optimization of the delay structure for form II. The cascade form implements a filter in a series of second-order IIR. This implementation explores the linearity of the filter and minimizes problems with the coefficient magnitude.

2.3.3 Decimation and Half-band filters

The decimation filters are composed of an anti-aliasing filter and decimation module. The idea is to remove frequency components above $f_s/(2L)$, where f_s is the sample rate and L is the decimation factor, because copies of the spectrum are generated after decimation, resulting in output distortions.

$$y[n] = x[Ln] \tag{9}$$

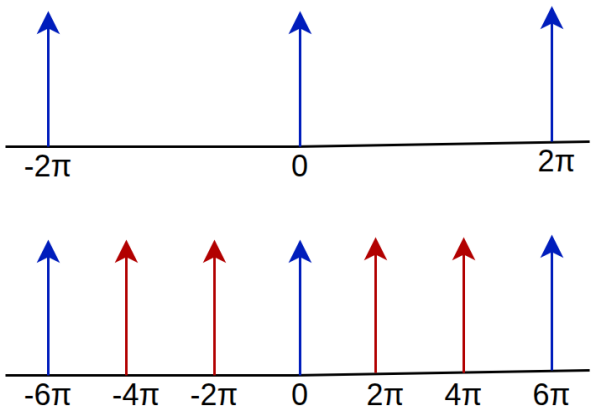
Figure 5 shows the effect of the decimation with $L = 3$. It is observed that the original component is dislocated by 4π, two copies are created, and the amplitude of the signals is reduced by the decimation factor.

A common approach to aliasing suppression involves half-band filters, which attenuate all signal components above $f_s/4$. These filters are typically combined with a decimation factor of 2. Implemented as FIR filters, they leverage coefficient symmetry to minimize resource utilization: the number of multipliers can be halved, and only one-quarter of the coefficients need to be stored due to their symmetric properties.

2.3.4 Filter Architecture Analysis

Figure 6 and Figure 7 show the magnitude and phase response for different filter architectures. The 4^{th} order IIR filter exhibits a magnitude response comparable to those

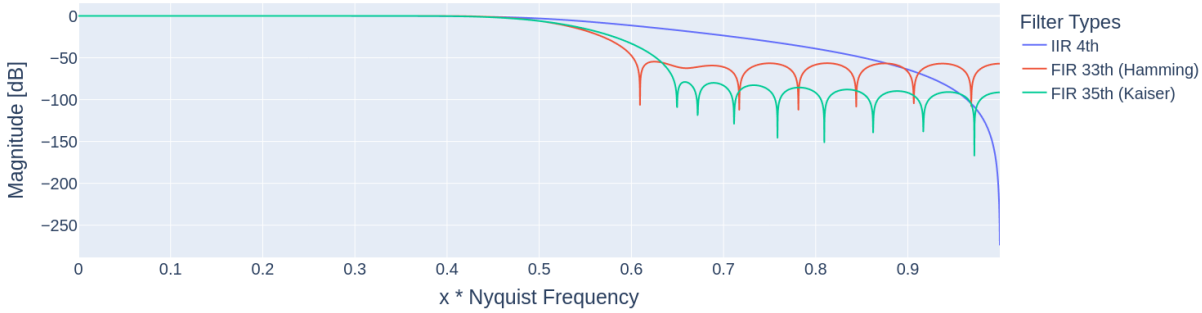
Figure 5 – Decimation effect in frequency domain



Source: (ULABY; YAGLE, 2018)

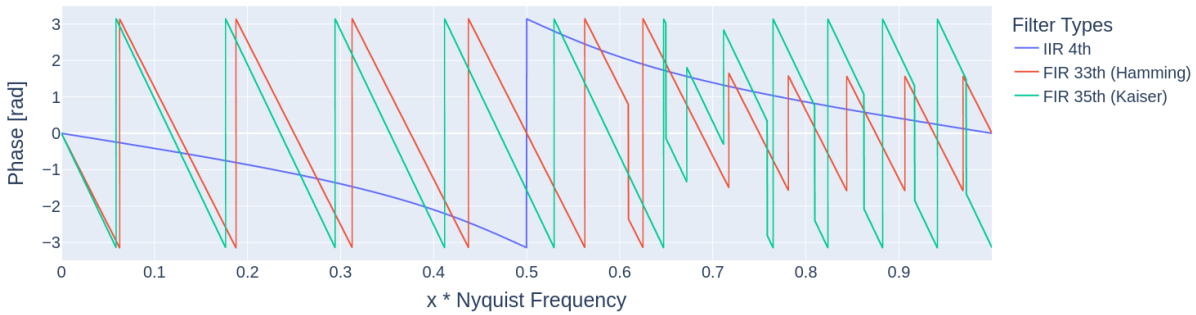
of the 33rd and 35th order FIR filters while requiring significantly fewer hardware resources. For an FIR filter of order N, each output sample requires N multipliers and N adders per iteration. In contrast, the IIR implementation (assuming a direct-form structure) typically uses $2N + 1$ multipliers and $2N + 1$ adders. By adopting a 4th order IIR instead of a 33rd or 35th order FIR, the design saves between 24 and 26 multipliers and the same number of adders, leading to a more efficient FPGA implementation.

Figure 6 – Magnitude Response: IIR, FIR (Hamming), FIR (Kaiser)



Phase distortion was not a critical concern in the LIA's low-pass filter design, as its primary purpose was to extract only the DC component of the signals. While FIR filters can meet the selectivity demands, they require significantly more hardware resources compared to an IIR filter of similar performance. This resource disparity becomes even more pronounced as the transition bandwidth between the passband and stopband narrows. A key challenge with IIR filters, however, arises when the cutoff frequency is much lower than the sampling rate. Under such conditions, the numerator coefficients (b) approach zero, requiring an more bits for correct representation.

Figure 7 – Phase Response: IIR, FIR (Hamming), FIR (Kaiser)



2.4 DECIMAL REPRESENTATIONS

There are different ways to represent decimal numbers in digital systems: fixed-point, in which the numbers are represented as integers, with bits divided into integer and fractional parts; and floating-point, with bits divided into a fractional part and an exponent. Figure 8 illustrates how the bits are interpreted. The fixed point is a signed Q7.23, the initial seven bits represent integer numbers $(2^7, 2^6, ..., 2^0)$, and the remaining bits are fractions of two $(2^{-1}, 2^{-2}, ..., 2^{-23})$. On the other hand, the floating representation of the fractional part, also called mantissa or significant, are fractions of two, and the other bits consist of the base exponent, to recover the number $significand * base^{exponent}$.

Figure 8 – Floating-point vs. Fixed-point bit format

IEEE 754 Single Precision Floating Point Number																															
S	Exponent (8 bits)								Fractions (23 bits)																						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
32-Bit Fixed Point (Q8.23)																															
S	Integer (8 bits)								Fractions (23 bits)																						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

Source: (ASLAN; MOHAMMAD; SALAMY, 2016)

Digital system arithmetic varies according to the decimal representation; in fixed-point, the operations are performed with integers, and the only point of attention is the number of bits resulting from each operation. In summary, for $QN.M + QX.Y = QZ.W$, the integer and fractional parts must be equal$(N = X, M = Y)$, and the result must include an additional integer bit to avoid overflow$(Z = X + 1)$. In multiplication, $QN.M \times QX.Y = QZ.W$, there is no format restriction, but the result contains the sum of integer and fractional bits from both numbers and one extra integer $(Z = N + X + 1, W = M + Y)$(YATES, 2007). In floating point addition, it is necessary to shift the operands

to have the same exponent, then the fractional parts are summed. In multiplication, the exponents are summed and the mantissas are multiplied(PARHAMI, 2010). These operations are illustrated in Equation 10 and Equation 11.

$$(\pm s1 \times b^{e1}) + (\pm s1 \times b^{e1}) = (\pm s1 \pm \frac{s2}{b^{e1-e2}}) \times b^{e1} \tag{10}$$

$$(\pm s1 \times b^{e1}) \times (\pm s1 \times b^{e1}) = \pm(s1 \times s2) \times b^{e1+e2} \tag{11}$$

As illustrated, the operations with floating point add a layer of manipulation to the basic operations, and the addition is inefficient when compared with fixed-point arithmetic. The floating point can represent a larger interval of numbers with the same number of bits but with relative precision, which means that the distance between two numbers is not always the same. To simplify the hardware and considering that fixed precision is sufficient when measuring voltages in a predetermined range, fixed-point arithmetic was used in the LIA design.

2.5 DIGITAL SYSTEM ERROR ANALYSIS

Both Fixed-Point (FP) and Floating-Point representations are constrained by the limited number of bits available to represent real numbers, which results in quantization errors that are propagated through subsequent arithmetic operations. Understanding how such errors accumulate is crucial for selecting an appropriate number formats that balance hardware resource usage with the required precision for the target application.

The fundamental premise in modeling accumulated error is that a fixed-point number can be interpreted as higher-precision value affected by a quantization error (SIMIĆ et al., 2022), as described in Equation 12:

$$x_{fp} = x + \epsilon(x) \tag{12}$$

Using this definition, the error introduced in elementary operations such as addition and multiplication can be analytically estimated. These are shown in Equation 13 and Equation 14, respectively.

$$\begin{aligned} z_{fp} = x_{fp} + y_{fp} = x + \epsilon(x) + y + \epsilon(x) \\ \text{then } \epsilon(z) = \epsilon(x) + \epsilon(x) \end{aligned} \tag{13}$$

$$\begin{aligned} z_{fp} = x_{fp} * y_{fp} = x * y + \epsilon(x) * y + y * \epsilon(x) + \epsilon(x) * \epsilon(y) \\ \text{then } \epsilon(z) = \epsilon(x) * y + y * \epsilon(x) + \epsilon(x) * \epsilon(y) \end{aligned} \tag{14}$$

LIA is based on sums and multiplication, and using these basic formulas, it is possible to estimate the error in the system. An alternative framework for interpreting error

in fixed-point computations uses *affine arithmetic* (FANG; RUTENBAR; CHEN, 2003). In this model, quantization error is expressed as $\epsilon(x) = 2^{-f_x}\varepsilon$,the maximum truncation error of a fixed-point representation, where f_x denotes the number of fractional bits used to represent x_{fp}, and $\varepsilon \in [-1, 1]$ is an unknown bounded noise symbol. The resulting error expression includes an additional term related with the result fractional bit width, $\phi = 2^{-f_z}\varepsilon_z$. This term becomes negligible if $\max(f_x, f_y) > f_z$, in addition, and if $f_x + f_y > f_z$, in multiplication.

$$c_{fp} = c + E_c, \quad E_c \leq 2^{-f_c} \tag{15}$$

$$x_{fp} = x_0 + \sum_{i=1}^{n} x_i\varepsilon_i + E_x, \quad E_x = x_{e0} + \sum_{i=1}^{m} x_{ei}\varepsilon_{ei} \tag{16}$$

$$E_{x_{fp} \pm y_{fp}} = E_x \pm E_y + \phi \tag{17}$$

$$E_{c_{fp} \cdot x_{fp}} = E_x \cdot c + E_c \left(x_0 + \sum_{i=1}^{n} x_i\varepsilon_i \right) + \phi \tag{18}$$

$$E_{bound} = B(e_0 + \sum_{i=1}^{m} e_i\varepsilon_i) = \sum_{i=1}^{m} |e_i| \tag{19}$$

This bounding operation has a conservative nature, which assumes that all sources of error simultaneously attain their maximum values, an increasingly unlikely event as the number of operations grows. Then to mitigate this effect, the probabilistic bound is introduce, where the bounding is obtained through a confidence interval, as shown in Equation 20. And as the terms are added to the error, it tends to a normal distribution according to the central limit theorem, as described in Equation 21. $B_\lambda(E_n)$ is calculated using the inverse Cumulative Density Function (CDF) of a Gaussian when $n > 3$ (FANG; RUTENBAR; CHEN, 2003).

$$prob(E_n \leq B_\lambda(E_n)) \geq \lambda \tag{20}$$

$$\frac{E_n}{\sqrt{n}\sqrt{variance(E_n)}} \rightarrow \mathcal{N}(0, 1) \tag{21}$$

Another method to evaluate the error considers the distribution of the error: first, it is necessary to find the error distribution of the input parameters, and then propagate the error using the formulas in Equation 22 and Equation 23, where X, Y, and Z are the probability density functions of the random variables x, y, and z(AKIYAMA et al., 2024).

$$Z(z) = X(x) \oplus Y(y) = \int X(z - y)Y(y)dy \tag{22}$$

$$Z(z) = X(x) \otimes Y(y) = \int \frac{1}{|y|} X(\frac{z}{y}) Y(y) dy \tag{23}$$

$$\epsilon(y) = \sum_i (\epsilon(x_i) \frac{\partial y}{\partial x_i}) \tag{24}$$

The propagated error is divided into amplification and injection errors. The first comes from the derivative of the operation and has the format of Equation 13 for sum, and the format of Equation 14 without the term $\epsilon(x) * \epsilon(y)$ for multiplication.The injection error originates from the truncation error. Finally, using these formulas, it is possible to model the error distribution and determine their limits.

The *affine arithmetic* method gives an estimation of the error bound, without requiring knowledge about the input distribution, only the amplitude, and the operations consists of additions and multiplications, which are easier to implement and demand fewer computational resources. Due to these points, this method was chosen to evaluate the error in this thesis

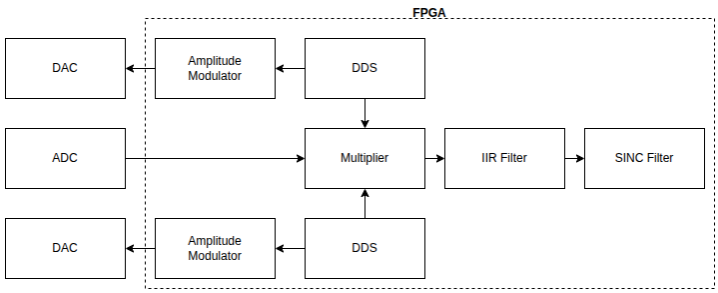
3 LITERATURE REVIEW

This section reviews various FPGA-based LIA implementations developed for different applications, highlighting their methodologies and key results. The objective is to identify common design strategies, performance trade-offs, and architectural innovations relevant to digital LIA systems. Based on this review, a compilation of the most relevant and effective ideas is presented, serving as a foundation for the design choices made in this work.

3.1 PATHOGEN DNA DETECTION

The authors presented a LIA design for pathogen DNA detection using magneto-resistive sensors that employs a sophisticated architecture centered around a Spartan-3 FPGA, which serves as the core of the embedded digital processing unit(GERVASONI et al., 2014). This system implements a unique double modulation scheme, incorporating both magnetic and electrical modulations to enhance signal detection. Two independent generation channels are utilized: one for electrical modulation, spanning from a few Hz up to 100 kHz, driven by a 16-bit resistor ladder Digital-to-Analog Converter (DAC); and another for magnetic modulation, operating from a few Hz up to 2 kHz, also driven by a DAC and an external power amplifier. To manage the 12 magneto-resistive sensors efficiently over long experimental timescales, a single acquisition chain is employed, featuring an input multiplexer that sequentially selects each sensor. The signal conditioning path includes a custom-designed differential amplifier for signal amplification and common-mode rejection, followed by an anti-aliasing filter and a 16-bit, 1 Msps ADC for digitization. The digital architecture within the FPGA performs waveform generation using Direct Digital Synthesizer (DDS) blocks and executes a cascaded demodulation process, involving two stages of multipliers at distinct frequencies, complemented by IIR and SINC filters for noise rejection and signal processing, as illustrated in Figure 9.

Figure 9 – Pathogen DNA detection: LIA Digital Implementation



Source: (GERVASONI et al., 2014)

The performance assessment of the developed platform demonstrated significant

improvements in noise performance compared to commercial lock-in systems. When measuring standard resistors, the custom instrument achieved a noise floor of 240 nV_{rms} over 200 seconds with a 3 Hz bandwidth, which is 11 times better than the Zurich Instruments HF2LI and 3.5 times better than the Stanford Research System SR830 under identical conditions. Further enhancement was observed with the use of low thermal coefficient resistors, reducing the noise to an impressive 45 $n\text{V}_{\text{rms}}$, corresponding to a resolution of 0.7 *ppm*. This high resolution is crucial for detecting subtle resistance variations in DNA detection, where the signal related to analyte concentration can be a small fraction of the total resistance change. The system's ability to track magnetic nanoparticle sedimentation in real-time was also successfully demonstrated, confirming its operational capability for biological applications. The dual lock-in architecture implemented in the FPGA, leveraging its 48 multipliers, contributed to a $3dB$ improvement in SNR compared to a single demodulation approach. These results highlight the platform's capability for high-resolution, low-noise detection in magneto-resistive biosensing applications, particularly for pathogen DNA detection.

3.2 SQUARE WAVE MODULATION

Jiang et al. propose a novel algorithm for a high-modulation frequency and high-speed digital lock-in amplifier designed to enhance performance in online systems(JIANG et al., 2015). The core of their approach lies in utilizing a square wave reference signal whose frequency is an odd sub-multiple of the modulation frequency. This design capitalizes on the inherent odd harmonic components within the square wave to facilitate signal processing. A critical structural element of this LIA is the meticulous selection of the sampling frequency, which is set at four times the modulation frequency. This specific ratio ensures the orthogonality of the reference sequences, a fundamental requirement for accurate phase-sensitive detection. Furthermore, a significant innovation in this algorithm is the complete elimination of multiplication operations in the phase-sensitive detection stage, relying solely on additions and subtractions. This simplification is intended to reduce computational burden and accelerate the processing speed within the lock-in amplifier.

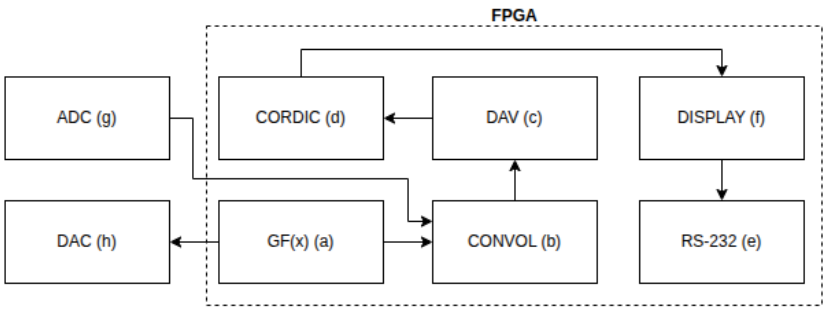
The efficacy of this new algorithm was rigorously tested through both simulations and experiments, demonstrating considerable improvements in the maximum modulation frequency achievable by the LIA. By replacing computationally intensive multiplication operations with simpler additions and subtractions, the algorithm significantly reduces the time consumption associated with signal processing. This reduction in computational burden, coupled with the ability to operate at a lower reference frequency relative to the modulation frequency, leads to a faster system response. The authors illustrate that their proposed LIA can effectively improve the maximum modulation frequency for a given reference frequency, as evidenced by comparisons of in-phase waveforms and frequency-domain components. This advancement is particularly relevant for real-time applications,

such as online cable capacitance measurement, where high modulation frequencies and rapid response times are critical. The results confirm the feasibility and advantages of this novel algorithmic approach in enhancing the speed and operational frequency range of digital lock-in amplifiers.

3.3 LOW-FREQUENCY APPLICATIONS

Macias-Bobadilla et al. present the development of a Digital Lock-In Amplifier specifically designed for low-frequency applications, leveraging the capabilities of FPGA(MACIAS-BOBADILLA et al., 2016). The core structure of this LIA is built upon a Spartan 3 FPGA, which facilitates a reconfigurable hardware design. Unlike analog LIAs that rely on expensive analog multipliers, this digital implementation utilizes optimized and internal multipliers within the FPGA, significantly reducing cost. The system's architecture, as shown in Figure 10, comprises several key modules: a Function Generation Module(a), which employs Look-Up Tables (LUT) to digitally store and generate sine waveforms with selectable offset, amplitude, and frequency; a Convolution Module(b) for performing the multiplication process; a Dynamic Average(c) Module that filters the signal; a Coordinate Rotation Digital Computer (CORDIC) Module(d) to calculate the module and phase; and Data Transmission(f,e) modules. This integrated FPGA-based design allows for the generation and selection of appropriate frequencies for various experiments, ranging from 1 Hz up to 10 kHz, depending on the ADC and DAC used.

Figure 10 – Low-Frequency Applications: LIA Digital Implementation



Source: (MACIAS-BOBADILLA et al., 2016)

The experimental results demonstrate the effectiveness of this FPGA-based LIA for low-frequency applications. The system exhibits good frequency stability, reported to be in the order of 1nHz, which is crucial for precise measurements. The authors highlight the linearity and repeatability of the response, comparable to commercial laboratory amplitude and phase shift detection devices, but with significantly lower error and standard deviation. This performance is particularly valuable for photothermal experiments, where important

activity often occurs at low frequencies (1 Hz to 10 kHz) and manifests as variations in amplitude or phase. The study emphasizes that this reconfigurable digital Phase Sensitive Detection system offers a more cost-effective and simpler implementation for low-frequency experiments compared to expensive commercial LIAs that often provide more complex functions than required. The ability to generate and select frequencies based on the specific experiment or material further enhances its versatility, making it suitable for online or permanent monitoring systems with minimal resource requirements.

3.4 WIDE-BANDWIDTH ELECTROCHEMICAL MEASUREMENTS

Carminati et al. introduce a compact FPGA-based elaboration platform designed for wide-bandwidth electrochemical measurements, which integrates a digital dual-phase lock-in demodulator(CARMINATI et al., 2012). The core of this system is a Xilinx Spartan-3 FPGA, specifically the XEM 3010 module, which manages communication with the host PC and various converters, enabling real-time data processing at an 80 MHz clock frequency. The LIA implemented within the FPGA incorporates a DDS for generating precise in-phase and in-quadrature reference signals. This structure also includes embedded hardware multipliers operating at 80 MHz for the necessary multiplication steps in lock-in detection. Subsequent low-pass filtering, including a third-order filter and an optional SINC filter, is employed for bandwidth reduction and decimation, allowing for highly customized measurement protocols and real-time data processing.

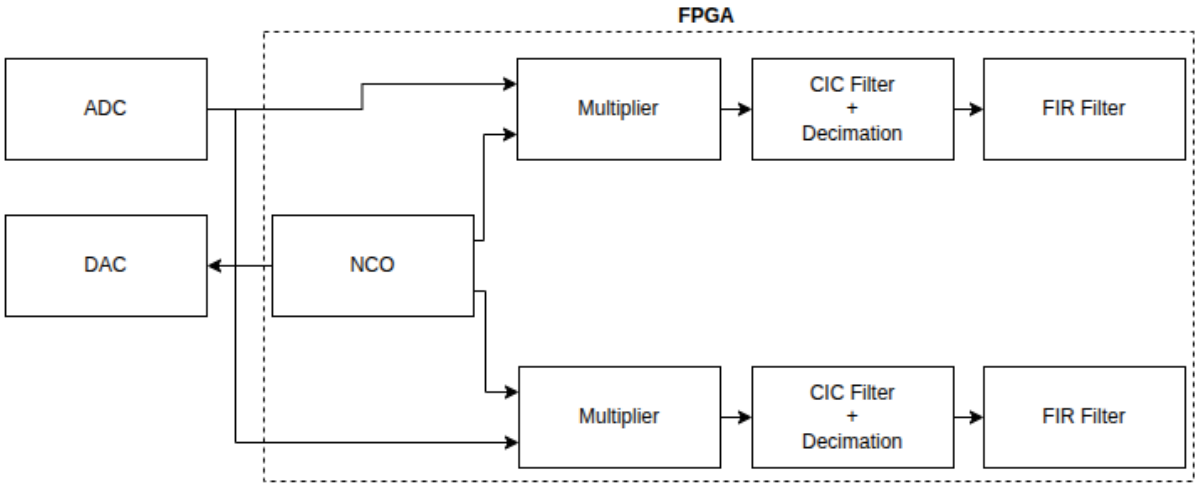
The experimental characterization of this platform demonstrates its capability for high-resolution impedance measurements across a wide spectral range, extending up to 10 MHz. When tested with an electrochemical dummy cell, the system achieved a full 10 MHz bandwidth with maximum errors of 5% in magnitude and 1° in phase, which are considered tolerable for electrochemical applications and can be further corrected with calibration. The platform also exhibits excellent noise performance; an impedance time tracking test revealed a noise level of 20 μV_{rms} at the lock-in output, resulting in a Signal-to-Noise Ratio (SNR) of 36 dB for 1% resistive pulses and 14 dB for 0.1% pulses. By reducing the bandwidth to its minimum achievable value, 777 Hz, the noise of the entire processing chain was reduced to approximately 1 μV_{rms}, representing a 126 dB reduction below the maximum input signal. These results highlight the platform's ability to provide performance comparable to bench-top and portable state-of-the-art instrumentation, while offering a compact and versatile solution for electrochemical sensing and other applications requiring fast and synchronous data generation and acquisition.

3.5 ELECTRICAL IMPEDANCE SPECTROSCOPY

Divakar et al. present a low-cost and portable FPGA-based lock-in amplifier specifically designed for measuring the dielectric properties of individual cells through electrical

impedance spectroscopy (DIVAKAR et al., 2018). The architecture of this LIA leverages the parallel processing and reconfiguration advantages of FPGAs, allowing for a compact design that significantly reduces the space, weight, and cost typically associated with conventional LIAs. The system integrates an ADC and DAC card with a data rate of 65 MSPS and 14-bit resolution. A key component is the Numerically Controlled Oscillator (NCO), which generates the reference signal (both sinusoidal and 90-degree phase-shifted cosine) using the CORDIC algorithm, operating at 65 MHz. The input signal from the microfluidic device is multiplied with these reference signals by a dedicated multiplier, and the resulting in-phase (X) and out-of-phase (Y) components are then processed through Cascaded Integrator-Comb (CIC) filters for decimation and a FIR filter for further noise reduction. The final outputs are saved via a USB-UART cable, as illustrated in Figure 11.

Figure 11 – Electrical Spectroscopy: LIA Digital Implementation



Source: (DIVAKAR et al., 2018)

The validation of this FPGA-based LIA demonstrates its excellent ability to suppress noise and accurately detect impedance changes in biological samples. The system successfully measured the electrical properties of individual Red Blood Cell (RBC) flowing through a microfluidic channel, even in a highly noisy environment. The experimental results show clear positive and negative peaks in the X and Y components of the measured signal, corresponding to the passage of single RBCs. Furthermore, the system demonstrated its capability to process a high throughput of cells, with impedance peaks for over 100 RBCs processed in 14 seconds. This high-throughput capability, combined with the noise suppression achieved through the phase-sensitive measurement principle, highlights the system's potential for non-invasive and label-free characterization of cellular dielectric properties. The successful integration of the LIA with a custom-built instrumentation amplifier and a microfluidic device provides a robust platform for real-time monitoring of cellular changes, offering valuable insights into the biological state of cells.

3.6 DESIGN COMMENTS

All architectures use a structure to synthesize the waveform inside the FPGA and stimulate the DUT through a system controlled by a DAC. The modulation signals are generated using DDS, except in the LIA for low-frequency applications, where a LUT is used. In this case, the waveforms are stored in memory and adjusted according to the measurement. The LUT-based approach appears simpler to implement and yields equivalent results; however, it likely requires more resources due to the need to store reference signals.

The square wave modulation LIA presents an alternative implementation of the LIA algorithm, but it also introduces a constraint to the system. It enforces a fixed relationship between the sampling frequency and the modulation wave frequency, meaning that when the modulation frequency (f_m) changes, the sampling frequency (f_s) must also be adjusted, following the relation: $f_s = 4h \cdot f_m, \quad 1 \leq h$ (JIANG et al., 2015). Moreover, as the main objective of this work is to understand and establish the foundations for future developments related to LIAs, it is more appropriate to begin with a traditional implementation using multipliers and a low-pass filter, as shown in the structure in Figure 11.

Another important consideration is the filtering process combined with decimation, as used in the electrical spectrometry LIA. This technique extends the system's operational frequency range while keeping hardware requirements manageable. As the difference between the sampling rate and the cutoff frequency increases, the filter coefficients tend to be smaller, demanding more hardware for implementation. By applying decimation, the sampling rate is reduced, and aliasing is prevented using appropriate filters. The trade-off of this approach is that the system's time constant increases proportionally with the decimation factor.

4 SYSTEM DESIGN

This section presents the architectural decisions and implementation methodology for the LIA system. The design exploration begins with software prototyping in Python. Two numerical precision approaches are then analyzed: a double-precision floating-point implementation serving as a reference model, followed by an optimized fixed-point version with detailed error quantification to balance precision and resource utilization. The Register-Transfer Level (RTL) synthesis outcomes demonstrate how these design choices translate into actual hardware performance, providing metrics for FPGA resource allocation. Finally, the final design is described, and the analog front-end components are explained.

4.1 DESIGN CHOICES

4.1.1 Phyton: Libraries and Functions

Pandas, Plotly, Numpy, and Scipy were the main libraries used for this project. Pandas were used to format and manipulate data, and their main role was to save results and make the operations with the data easier. Numpy offers general operations such as module, square-root, and options to generate waves, and was applied in general calculations and the generation of test signals. Plotly is a plot library that creates interactive plots, making the result analysis easier. Scipy was the most important, because it offered the functions to project the filters used in the circuit.

The functions $iirfilter$ and $firwin$ from Scipy were applied in the filter design. Function $iirfilter$ receives the filter order, cutoff frequency, sample rate, band type (low pass, high pass), type of IIR to design (Butterworth, Chebyshev, Elliptic), and output form. Other parameters are also available, such as the maximum ripple in the passband and the minimum attenuation in the stopband, but only when the filter is elliptical or Chebyshev. The output form has three options: second-order sections, numerator/denominator, and pole zero. The last two options are similar, providing the coefficients b and a specified in Equation 7, whereas the first option returns the coefficients for a cascaded implementation. Function $firwin$ has similar inputs: filter order, cutoff frequency, sample rate, and band type (low-pass and high-pass). It also has the option of applying a window to the resulting filter, such as Hamming or Kaiser(SCIPY COMMUNITY, 2024).

4.1.2 LIA Implementation: Double Precision

Considering the hardware available for the implementation is a DE10-Nano, that has an integrated ADC, LTC2308, with 12 bit resolution and 73.3 dB Signal-to-Noise and Distortion Ratio (SINAD)(ANALOG DEVICES, INC., 2016). Based on this information, the Effective Number of Bits (ENOB) is 11.88, obtained from Equation 25, then the

minimum voltage variation detected by the ADC is approximately $1mV$. Thereby, to design the LIA, the main effort was to keep an error smaller than 1mV, to use the full hardware potential.

$$ENOB = (SINAD - 1.76)/6.02 \tag{25}$$

The reference signal is generated with a DAC, MCP4725, which has a resolution of 12 bits, and I2C communication(MICROCHIP TECHNOLOGY INC., 2021). Owing to restrictions in the I2C communication frequency, the signal has a limited sample rate of 52kHz, which is used to generate the signals in the first LIA implementations.

The initial implementations used double precision floating point, and concerns about the precision of variables were eliminated and only considered in following steps. Two architectures of the LIA were analyzed: the first used a 4^{th} order IIR and the second a 1025^{th} order FIR, with a cutoff frequency of 100 mHz. The reference wave was a 2kHz sinoide, and the objective was to recover the amplitude. In the first test with FIR, using only the input signal without noise, the variation in the output was greater than the desired millivolt precision. Then, considering this fact, the FIR architecture were discarded, because increasing the order would require excessive hardware resources.

```
    fs = 52e3    #Sample rate
    f = 2e3      #Input Signal Frequency
    t = 2        #Time Interval
    s, ref, qref = generate_reference(t, fs, f)

    len_signals = 50
    v1 = signal_amplitude*(np.cos(2*np.pi*f*s)) + 2.5
    ref = np.array(list(ref[2:28])*2520*len_signals)
    qref = np.array(list(qref[2:28])*2520*len_signals)

    signal_amplitude = 0.3  #Input Signal Amplitude
    np.random.seed(42)      #Fixed seed for noise generation

    #Generate IIR Filter
    cutoff = 0.1
    order = 4
    sos = signal.iirfilter(order, cutoff, btype='lowpass', ftype=
       ↪ 'butter', fs=fs, output='sos')

    # Evaluate the system for different noise levels
    for amplitude in [0.1, 0.3, 0.5, 1, 2, 4, 8, 16, 32]:
        #Generate Input Signal
        n = amplitude*np.random.normal(0, 1, len(s))
```

```
v = np.array(list(np.array(list(v1[5:31])*2520) + n
    ↪ [:65520]) * len_signals)

#Multiply Input by Reference signals
mult = v*ref
qmult = v*qref

#Filter Multiplication Result
out_iir = iir_filter_float(mult, 2, sos)
out_iir_q = iir_filter_float(qmult, 2, sos)

#Calculate Module and Phase
mod_float = ((np.array(out_iir)**2 + np.array(out_iir_q)
    ↪ **2)**0.5)*2
pha_float = np.atan(np.array(out_iir_q)/(np.array(out_iir
    ↪ ) + 0.000001))
```

In the code snippet, only one period of the reference and input signals is replicated to simulate the behavior of the synthesized circuit, in which one period is saved in memory and used in the calculations. The choice of 65520 in the noise is related to the memory size limitation that can be implemented in the FPGA using the IP available in Quartus. In Table 3, the results from the Python code are presented, and the standard deviation in the module and phase tends to increase with a reduction in the SNR, as expected; however, the deviation is still below the acceptable value. The reference values in the example would be a 300mV module and 0.72 *rad* phase, as the SNR decreases, the mean values deviate from the expected value, which probably is a result of the noise component at 2kHz, which cannot be removed.

Table 3 – Signal Characteristics at Various Noise Amplitudes

SNR [dB]	**Module [V]**	**Phase [rad]**
6.515	0.299 ± 5.006e-05	-0.723 ± 7.501e-07
-2.988	0.302 ± 5.045e-05	-0.723 ± 1.368e-06
-7.429	0.296 ± 4.953e-05	-0.725 ± 5.364e-06
-13.473	0.295 ± 4.945e-05	-0.738 ± 3.703e-06
-19.492	0.306 ± 5.137e-05	-0.671 ± 1.769e-05
-25.526	0.284 ± 4.747e-05	-0.600 ± 3.791e-05
-31.528	0.2975 ± 5.864e-05	-0.967 ± 2.747e-05
-37.549	0.398 ± 7.262e-05	-0.821 ± 2.218e-05
-43.577	0.411 ± 1.048e-04	-0.800 ± 1.162e-04

To verify the impact of the 2kHz noise component, a simulation of the noise seed was conducted, and the results are presented in Figures 12 and 13. The results are recovered using the lowest SNR presented in Table 3. While the mean of the magnitude estimates converges to the correct 300 mV amplitude as the number of simulations increases, the phase estimation does not exhibit similar convergence behavior.

Figure 12 – Double Floating Point LIA: Module Mean Value for 20 Different Noise Seeds

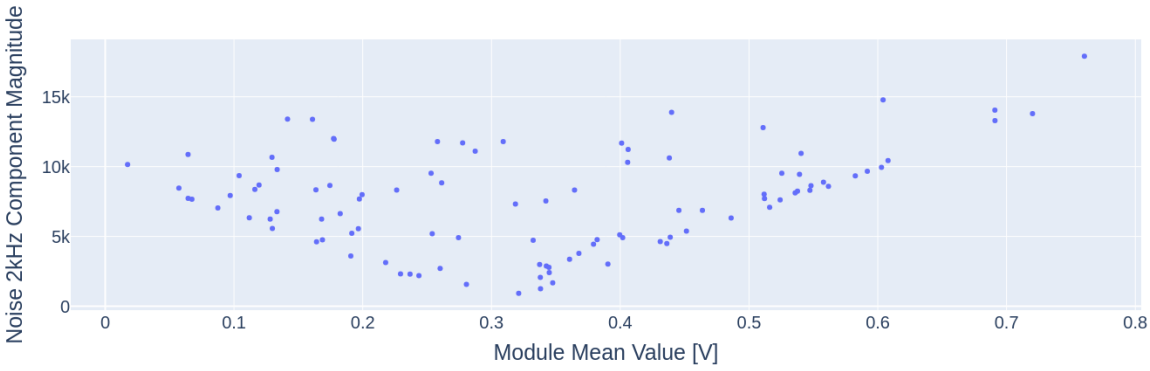
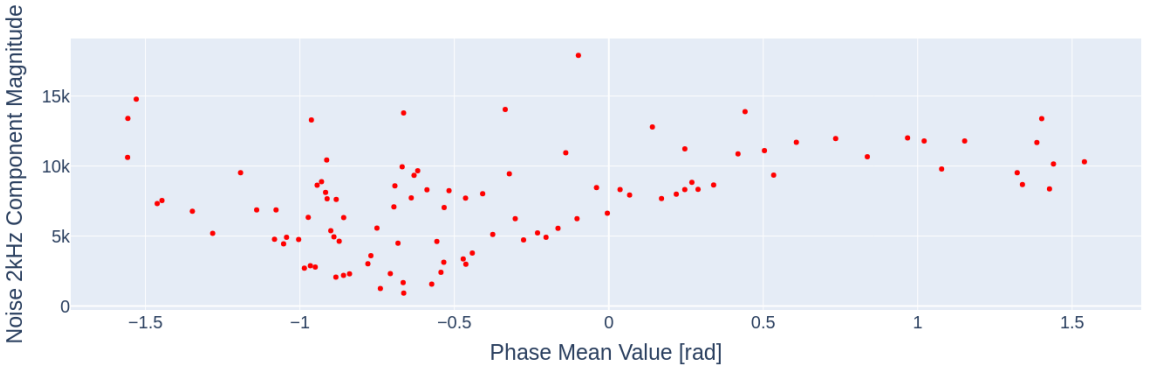


Figure 13 – Double Floating Point LIA: Phase Mean Value for 20 Different Noise Seeds



Another analysis that can help understand the performance of the LIA is to verify how the standard deviation is modified according to the noise seed. In Figures 14 and 15, the standard deviation remains inside the limit of 1mV in all scenarios. For this phase, the precision remained at 1 mrad in 97% of the scenarios.

Figure 14 – Double Floating Point LIA: Module Standard Deviation for 100 Different Noise Seeds

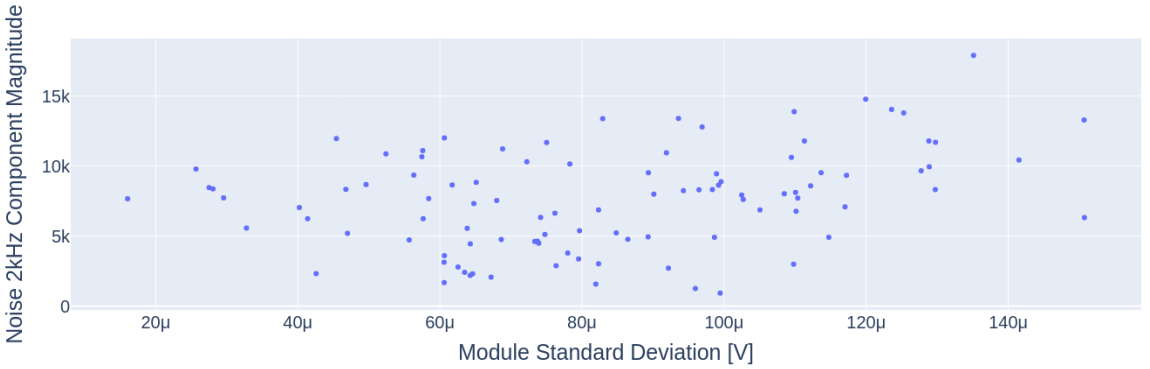
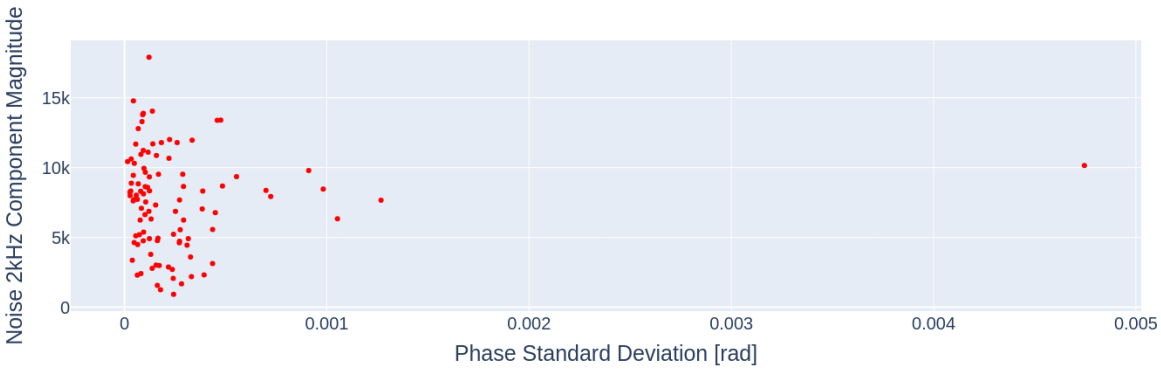


Figure 15 – Double Floating Point LIA: Phase Standard Deviation for 100 Different Noise Seeds



4.1.3 LIA Implementation: Fixed-Point

The next step was a fixed-point implementation using the same circuit architecture. The ENOB from the ADC was used as a reference to choose the precision of the input fixed-point. Considering that only a variation of 1mV can be detected, 10 fractional bits were used, $2^{-10} = 0.9765 * 10^{-3}$, and three integer bits were used to represent a range from 0 to 4V. The reference waves had an initial format of signed $Q1.14$ because they consisted of cossine and sine waves with an amplitude of one.

The filter coefficients required sufficient precision to represent the smallest value resulting from the design using the function $iirfilter$, which was on the order of 2^{-21}. Initially, 75 fractional bits were chosen, $2^{-75} = 2.65 * 10^{-23}$, and the same precision was used for the internal operations of the filter. To compare the double precision and fixed-point implementations, the architectures were tested without noise.

Table 4 – Comparison of Double and FP Measurements at Various Signal Amplitudes

Amplitude [V]	**Double Module [V]**	**FP Module [V]**	**Double Phase [V]**	**FP Phase [V]**
1.2	1.200 ± 2.01e-04	1.187 ± 1.99e-04	0.000 ± 2.70e-08	-0.003 ± 1.18e-08
1.0	1.000 ± 1.67e-04	0.987 ± 1.65e-04	0.000 ± 3.17e-08	-0.004 ± 3.63e-08
0.5	0.500 ± 8.36e-05	0.490 ± 8.22e-05	0.000 ± 6.01e-08	-0.008 ± 7.79e-08
0.3	0.300 ± 5.02e-05	0.293 ± 4.91e-05	0.000 ± 9.80e-08	-0.013 ± 3.16e-07
0.15	0.150 ± 2.51e-05	0.144 ± 2.39e-05	0.000 ± 1.93e-07	-0.026 ± 2.23e-07
0.1	0.100 ± 1.67e-05	0.094 ± 1.57e-05	0.000 ± 2.88e-07	-0.041 ± 2.54e-06
0.075	0.075 ± 1.26e-05	0.070 ± 1.22e-05	0.000 ± 3.82e-07	-0.054 ± 5.30e-07
0.05	0.050 ± 8.36e-06	0.044 ± 7.28e-06	0.000 ± 5.72e-07	-0.087 ± 7.12e-07
0.03	0.030 ± 5.02e-06	0.024 ± 3.31e-04	0.000 ± 9.51e-07	-0.157 ± 2.03e-03

Based on the results in Table 4, there is a difference in the mean values of the module and phase, which is caused by the quantization required in FP arithmetic. To analyze this effect and determine the main source of this error, the affine arithmetic technique was applied, considering the architecture of the system.

4.1.3.1 Fixed-Point Error Analysis

The first operation is multiplication, $mult$, between the input signal, in, and reference signal, ref. Equation 26 gives the error bound for the operation, considering that A_{in0} and A_{ref0} are the waves offset, and A_{in} and A_{ref}, the amplitudes. The fixed-point values are truncated, resulting in an error of 2^{-f} then, $E(in) = 2^{-10}\varepsilon_{ein} + e_{in0}$ and $E(ref) = 2^{-14}\varepsilon_{eref} + e_{ref0}$. Based on this, the term $(A_{ref0} + A_{ref}) * E(in)$ dominates the error because $(A_{in0} + A_{in})$ is not higher than 4, and $E(ref) * E(in)$ is insignificant.

$$\begin{aligned} mult = in \times ref = [A_{in0} + A_{in}\varepsilon_{in} + E(in)] \times [A_{ref0} + A_{ref}\varepsilon_{in} + E(ref)] \\ E(mult) = (A_{in0} + A_{in})E(ref) + (A_{ref0} + A_{ref})E(in) + E(ref)E(in) \end{aligned} \tag{26}$$

The filter follows a similar mathematical structure, implemented as a sum of products, but exhibits amplitude variations at each iteration due to its recursive nature. The general form of the IIR filter's error propagation is derived in Equation 27 where $\phi = 2^{-75}$ represents the truncation error introduced by storing intermediate results with 75 fractional bits. The total error $E(x)$ comprises two components: the propagated error from multiplicative operations and the fixed quantization error of the IIR coefficients. Since some error terms depend recursively on previous outputs, multiple iterations are required to compute a stable error bound.

$$\begin{aligned} y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2] - a_1y[n-1] - a_2y[n-2] \\ E(x[n-i]) = A_xE(b_i) + b_iE(x) + E(x)E(b_i) + \phi \\ E(y[n-i]) = A_yE(a_i) + a_iE(y-i) + E(a_i)E(y_i) + \phi \\ E(y[n]) = E(x[n]) + E(x[n-1]) + E(x[n-2]) - E(y[n-1]) - E(y[n-2]) \end{aligned} \tag{27}$$

The CORDIC algorithm was applied to calculate the module and phase from the filter output. It also consists of a sequence of sums and subtractions with multiplications to obtain the module. The phase is obtained using sums and subtractions of tabled values; therefore, the main source of error is not the fixed-point representation. Equations 28 and 29 show error propagation in the algorithm. In this case, the error bound is associated with the number of iterations necessary to obtain the module.

$$\begin{aligned} x[n] = x[n-1] - d_i * y[n-1] * 2^{-i} \\ E(x[n]) = E(x[n-1]) - d_i * E(y[n-1]) * 2^{-i} \\ y[n] = y[n-1] + d_i * x[n-1] * 2^{-i} \\ E(y[n]) = E(y[n-1]) + d_i * E(x[n-1]) * 2^{-i} \end{aligned} \tag{28}$$

$$\begin{aligned} Module = Kx[n] \\ E(Module) = E(x[n])K + A_{x[n]}E(K) + E(x[n])E(K) \end{aligned} \tag{29}$$

By implementing the equations in Python, it is possible to obtain the error bound and the actual error for each amplitude presented in Table 4, and the results are presented in Table 5.

Table 5 – Comparison of Error Bound and Actual Error for Various Signal Amplitudes

Amplitude [V]	**Module Error [V]**	**Error Bound [V]**
1.2	1.283×10^{-2}	2.946×10^{-2}
1.0	1.305×10^{-2}	2.667×10^{-2}
0.5	9.763×10^{-3}	1.972×10^{-2}
0.3	6.911×10^{-3}	1.694×10^{-2}
0.15	5.898×10^{-3}	1.485×10^{-2}
0.1	5.539×10^{-3}	1.415×10^{-2}
0.075	4.589×10^{-3}	1.381×10^{-2}
0.05	5.888×10^{-3}	1.346×10^{-2}
0.03	5.622×10^{-3}	1.318×10^{-2}

Using the 300 mV amplitude as a test case, Table 6 summarizes the error bounds for each arithmetic operation. To meet the target precision of 1 mV, the minimum number of fractional bits required was systematically determined across all processing stages. This analysis was extended in Table 7, which evaluates error bounds for different bit-width configurations. Since the error scales with input amplitude, the worst-case scenario (using the ADC's full-scale range) was employed to optimize the fractional bit allocation. The results demonstrate that allocating 14 fractional bits to both the input and reference signals, combined with 85 fractional bits for the filter coefficients, yields an error bound below the 1 mV requirement.

Table 6 – Error and Error Bound for each operation with 300mV Input Signal

Operation	**Error**	**Error Bound**
Multiplication In Phase	9.307×10^{-4}	1.120×10^{-3}
Filter In Phase	3.468×10^{-3}	8.468×10^{-3}
Multiplication Quadrature	6.355×10^{-4}	1.120×10^{-3}
Filter Quadrature	1.933×10^{-3}	8.468×10^{-3}
Module CORDIC	6.911×10^{-3}	1.694×10^{-2}

Table 7 – Error Bound for Various Fixed-Point Precision

Fractional Bits			**Error Bound**		
Input	**Reference**	**Filter**	**Multiplication**	**Filter**	**CORDIC**
14	14	75	3.110×10^{-4}	1.872×10^{-2}	3.744×10^{-2}
14	14	80	3.110×10^{-4}	8.700×10^{-4}	1.740×10^{-3}
14	14	85	3.110×10^{-4}	3.222×10^{-4}	6.443×10^{-4}
16	16	75	7.776×10^{-5}	1.848×10^{-2}	3.697×10^{-2}
18	18	75	1.944×10^{-5}	1.842×10^{-2}	3.685×10^{-2}
20	20	75	4.860×10^{-6}	1.841×10^{-2}	3.682×10^{-2}
14	16	80	1.236×10^{-4}	6.824×10^{-4}	1.365×10^{-3}
12	14	85	4.941×10^{-4}	5.053×10^{-4}	1.011×10^{-3}

4.2 RTL SYTHESIS AND ANALYSIS

The design was synthesized using the best architecture from the last section, with the exception of the CORDIC module. The CORDIC Intellectual Property (IP) Module from Altera has a limit of 62 fractional bits in its input. This limitation did not reduce the precision of the result because all calculations inside the filter are performed with the desired number of bits, and only the outputs are truncated.

LIA was programmed in a DE10-Nano using Quartus Lite, the input signals were generated with Python, and inputted into the system using a RAM and the tool In-System Memory Content Editor. This tool enables modification of RAM data while the program is running on the FPGA. To recover the signals from the FPGA, another tool, the Signal Tap Logic Analyzer, was used, which externalizes signals in an integer format from different parts of the circuit. Finally, the data extract was analyzed and manipulated using Python and the libraries mentioned previously.

Table 8 compares the emulation and the results obtained in Python using the double-precision floating point. After modifications to the input signals and filter fractional bits, the error for all tested amplitudes was lower than 1 mV, and the phase error was also lower than 0.01 rad.

Table 8 – Comparison between Emulation and Python Results for Various Amplitudes

Python [Double]			**Emulation [FP]**	
Amplitude [V]	**Module [V]**	**Phase [rad]**	**Module [V]**	**Phase [rad]**
1.2	1.200 ± 2.76e-06	0.4833 ± 4.18e-09	1.199952 ± 6.43e-08	0.483294 ± 3.88e-12
1.0	1.000 ± 2.30e-06	0.4833 ± 2.54e-09	0.999962 ± 5.18e-09	0.483284 ± 1.08e-12
0.5	0.500 ± 1.15e-06	0.4833 ± 1.14e-09	0.499981 ± 3.84e-16	0.483287 ± 6.29e-16
0.3	0.300 ± 6.88e-07	0.4833 ± 7.00e-09	0.299968 ± 7.34e-14	0.483288 ± 4.00e-15
0.15	0.150 ± 3.45e-07	0.4833 ± 4.11e-09	0.149989 ± 1.28e-11	0.483241 ± 2.90e-14
0.1	0.100 ± 2.30e-07	0.4833 ± 5.09e-09	0.099990 ± 7.86e-12	0.483252 ± 4.21e-13
0.075	0.075 ± 1.72e-07	0.4833 ± 7.63e-09	0.074994 ± 9.70e-15	0.483306 ± 1.54e-13
0.05	0.050 ± 1.15e-07	0.4833 ± 1.16e-08	0.049974 ± 8.16e-15	0.483233 ± 8.39e-14
0.03	0.030 ± 6.91e-08	0.4833 ± 1.94e-08	0.029979 ± 8.18e-15	0.483265 ± 1.31e-13

The measured module error was on the order of 10^{-5}, approximately one order of magnitude larger than the theoretical error bound predicted in Table 5. This discrepancy suggests potential for further hardware optimization. Since multiplication errors were tightly bounded using affine arithmetic and limited by the CORDIC IP's fractional bit-width, the optimization efforts were on the IIR filter.

For evaluation,the Python implementation was modified to test the filter with varying fractional bit-widths. The test used a 2 kHz sinusoidal input signal with 2.048 V amplitude—the ADC's maximum input range—to produce worst-case error conditions. The selection of this maximum amplitude ensures the error analysis covers the system's full dynamic range.

Table 9 reveals that 79 fractional bits constitute the minimum bit width maintaining the fixed-point implementation error below the 1 mV threshold, as determined by

Table 9 – Filter Output Error for Different Fractional Bit Widths

Fractional Bits	**Mean Error**
70	3.393×10^{-1}
75	8.972×10^{-3}
76	5.250×10^{-3}
77	2.384×10^{-3}
78	1.356×10^{-3}
79	5.922×10^{-4}
80	3.115×10^{-4}
85	5.187×10^{-5}
90	4.688×10^{-5}

comparing double-precision floating-point and fixed-point IIR filter outputs. The analysis considered only mean error values, since the standard deviation was found to be three to four orders of magnitude smaller (10^{-3} to 10^{-4}) and thus negligible. Following IIR filter optimization, hardware emulation with a 2.048 V amplitude test signal demonstrated a final error of 0.98 mV, successfully validating the design meets its precision requirements while maintaining hardware efficiency.

Table 10 – Comparison between Emulation and Python Results after the optimization

Python [Double]			**Emulation [FP]**	
Amplitude [V]	**Module [V]**	**Phase [rad]**	**Module [V]**	**Phase [rad]**
2.048	2.048 ± 3.42e-4	0.4833 ± 9.72e-9	2.047017 ± 1.83e-09	0.483318 ± 4.50e-11
1.2	1.200 ± 2.76e-06	0.4833 ± 4.18e-09	1.199187 ± 5.6521e-12	0.483268 ± 2.47e-12
0.075	0.075 ± 1.72e-07	0.4833 ± 7.63e-09	0.074503 ± 4.65e-12	0.481567 ± 1.36e-12

To investigate the hardware saved after the optimization, the data from Quartus compilation report was used, specifically, the total Adaptative Logic Modules (ALM) and registers as shown in Table 11. The registers reduced in 4.83% and the ALMs reduced in 0.13%.

Table 11 – Compare hardware resources for both fractional bit widths

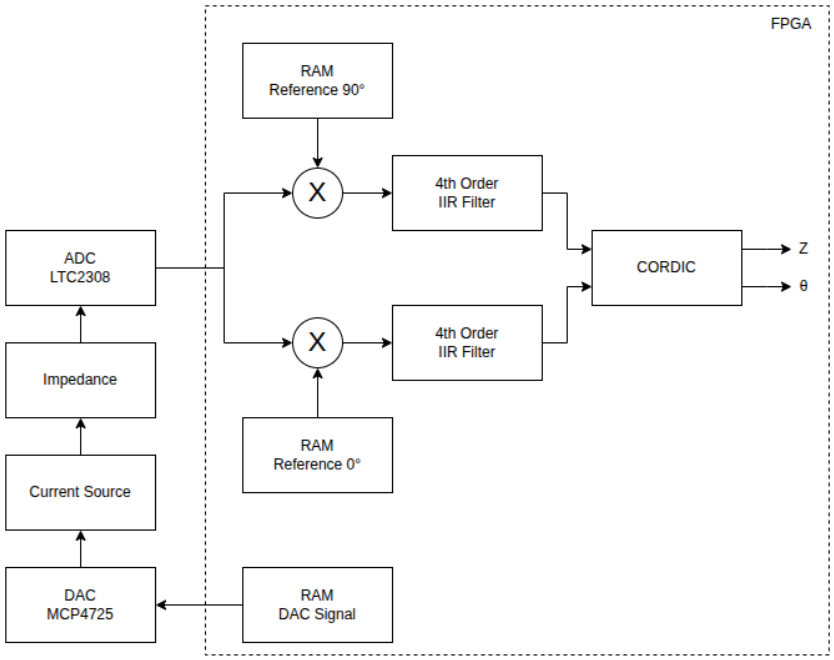
	Fractional Bit Width	
Resource	**79**	**85**
ALM	20443	21479
Registers	35742	35788

The reduction of 5 fractional bits yields hardware savings due to several factors. First, the IIR filter is implemented using second-order sections, with the module instantiated four times in the design. Second, fixed-point multiplications require storing results with a bit width equal to the sum of the operand widths, so reducing operand precision directly lowers resource usage. Finally, the reduced word length also simplifies the adders, further contributing to a smaller hardware footprint.

4.3 FINAL DESIGN

The final lock-in design is illustrated in Figure 16. The components implemented as digital logic within the FPGA are highlighted in the dashed region. Reference signals at 0° and 90° are stored in RAM blocks generated using Quartus IP cores, which also enable real-time debugging via the In-System Memory Content Editor. The incoming signal, digitized by the ADC (LTC2308), is multiplied by both reference signals to extract the in-phase and quadrature components. These products are then passed through 4^{th} order IIR filters to suppress noise. The filtered signals are processed by a CORDIC module to compute the magnitude (Z) and phase (θ) of the impedance. The CORDIC algorithm iteratively reduces one of the signal components toward zero and derives the phase based on the angle of rotation, while the magnitude is computed from the remaining component.

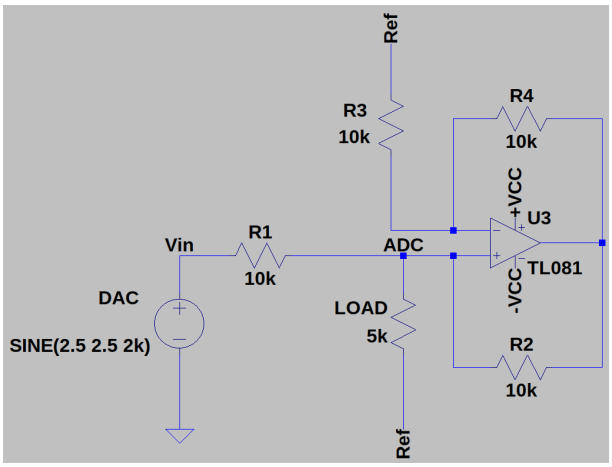
Figure 16 – Block Diagram: LIA Design



The DUT signal is generated by a DAC (MCP4725), allowing full control over the waveform characteristics. The analog front-end is shown in Figure 17, where a non-inverting amplifier is used to generate a known current through the load. The current flowing through the load resistor is defined by the voltage at the amplifier input and the value of resistor R_1, as described by Equation 30. This current, along with the measured voltage across the load, is used to compute the impedance magnitude after the LIA determines the voltage amplitude.

$$I_{load} = \frac{V_{inp}}{R_1} \tag{30}$$

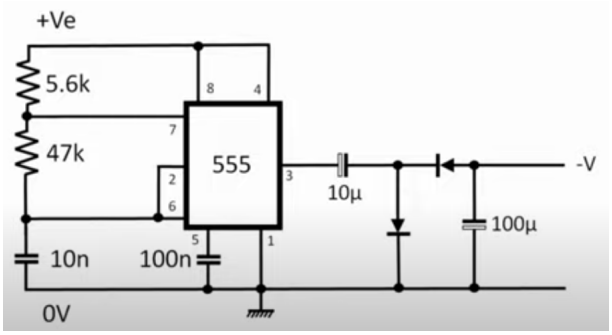
Figure 17 – Current Source Circuit



The TL081 operational amplifier used in the analog front end requires a dual supply, including a negative voltage for the $-V_{CC}$ rail. To generate this voltage, the circuit shown in Figure 18 was implemented using a 555 timer-based charge pump. The $+V_{CC}$ and REF voltages were sourced directly from the FPGA's onboard 9V supply pin, making the system more self-contained and independent of external power sources.

In this setup, the FPGA 9V pin was used both as $+V_{CC}$ and as the $+V_e$ input to the negative voltage generator, which produced a –6.8V rail. The negative voltage source, while functional, was less stable than the regulated FPGA voltage lines and introduced additional ripple and noise. The REF was set to 2.5V, to explore the resolution of the ADC, and generated from the same 9V supply using a resistor divider followed by a buffer to ensure low impedance and voltage stability.

Figure 18 – Negative Voltage Source Circuit



Source: (ELECTRONICS, 2021)

5 CHARACTERIZATION AND RESULTS

This section begins by presenting several key performance metrics, including input noise, system error across different SNRs, dynamic reserve, sensitivity and vertical resolution. A comparison is also made between an commercial LIA solution. Finally, the section concludes with a summary of the main design outcomes and insights.

5.1 PERFORMANCE METRICS

The LIA performance was evaluated using the metrics suggested by Zurich Instruments, a reference company in the electronics instrumentation area. Then, the frequency range, sampling rate, vertical resolution, sensitivity, input noise voltage, dynamic reserve and the demodulation bandwidth were analyzed(CIARDI, 2023). The demodulation bandwidth is defined by the 4^{th} order IIR filters, that have a cutoff frequency of 100mHz, as demonstrated in previous sections.

The system's frequency range and sampling rate were constrained primarily by the communication speed with the DAC. The FPGA communicates with the MCP4725 using the I2C protocol, whose transaction structure is illustrated in Figure 19. Each data transmission requires 33 clock cycles to complete. While the standard maximum I2C clock rate supported by the MCP4725 is 400kHz—corresponding to a theoretical sampling rate of approximately 12.1kHz—it was possible to increase the bus clock to 1.7MHz in practice, achieving a sampling rate of up to 52kHz.

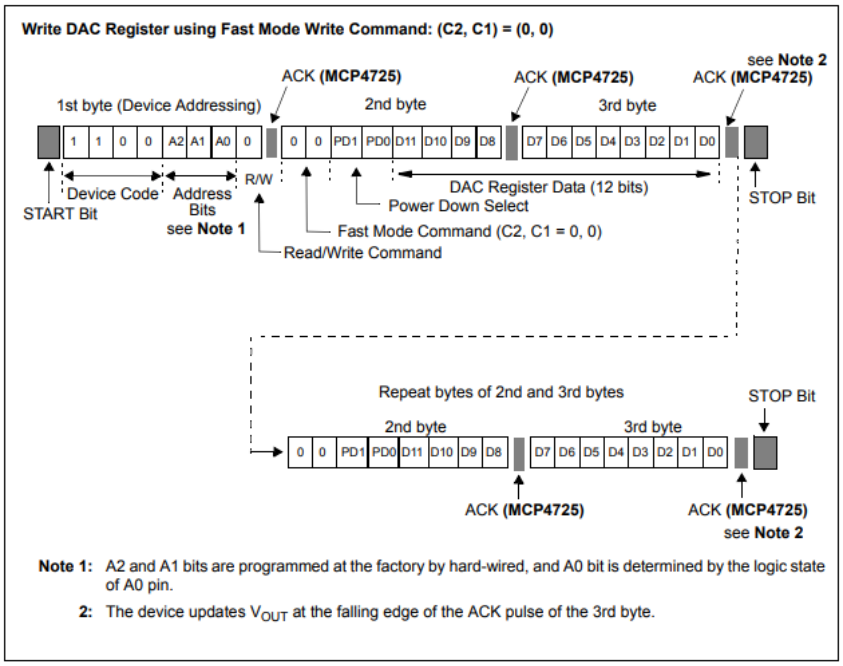
The vertical resolution is limited by the number of bits in the ADC, which in this case was 12 bits, the maximum variation registered, considering that the supply is fixed in 4.096V was 1mV. The sensitivity can be increase using dithering(POTTS, 2022), nevertheless in this case this technique was not applied, and was left as a future improvement to the design, then the sensitivity is equal to the vertical resolution.

5.2 INPUT NOISE

The input noise refers to the noise introduced by the system itself into the measured signal. This parameter was evaluated in terms of noise amplitude spectral density, expressed in $V/\sqrt{Hz}$. The measurement procedure involved performing a frequency sweep with different input signal amplitudes. The noise was extracted from the output of the digital filter by calculating the standard deviation of the output signal and dividing it by the square root of the demodulation bandwidth, which in this case was 100mHz.

Due to the number of frequency points and required reference signal updates, the measurement process is computationally intensive and repetitive. To automate this task, the ARM Hard Processor System (HPS) integrated in the DE10-Nano board was used. An Arch Linux system was installed on the MicroSD card to configure the HPS-FPGA

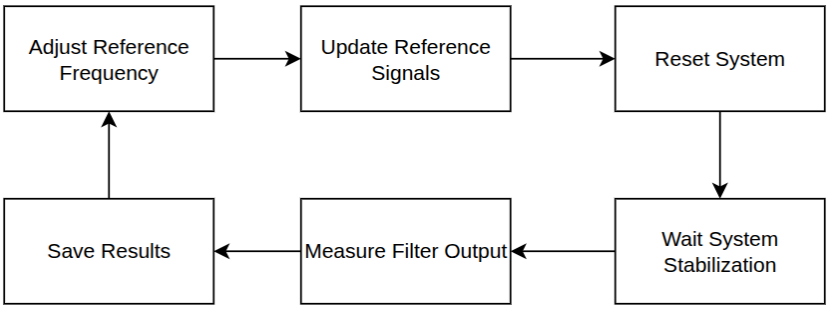
Figure 19 – MCP4725 Write Command Byte Structure



Source: (MICROCHIP TECHNOLOGY INC., 2021)

communication and allow execution of C scripts for measurement control. Intel Quartus provides a dedicated IP block to implement a memory-mapped bridge between the HPS and the FPGA, requiring only the definition of control signals and data export paths to complete the interface.

Figure 20 – Input Noise Measurement Pipeline

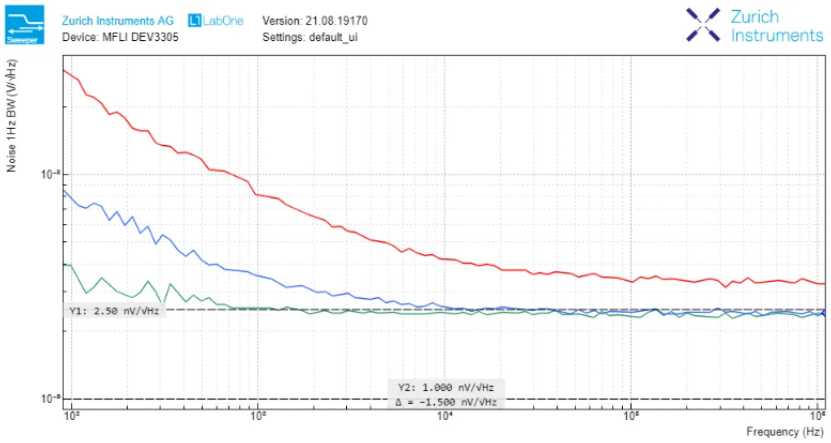


The measurement routine illustrated in Figure 20 was used to evaluate the system's input noise across different frequencies. The frequency is defined by the number of points in the reference signal stored in RAM. To change the frequency between measurements, the system adjusted the maximum value of the address counter that accesses the signal samples in RAM. In parallel, the contents of the RAM blocks for the DAC output, as well as the in-phase and quadrature reference signals, were updated accordingly.

After updating the reference signals, the system was reset and allowed to stabilize before data was collected. Once stable, the output of the digital filter was sampled and stored in a separate RAM block at a rate of 520Hz. This sampling rate was sufficient to capture the noise within the demodulation bandwidth while keeping acquisition time reasonable. Each frequency point was measured 5 times to improve statistical confidence, and the results are saved to a MicroSD card in CSV (Comma Separate Values) format for later analysis.

The ideal input noise spectrum for a Lock-In Amplifier would consist predominantly of flicker noise, also known as 1/f noise, without significant contributions from broadband or white noise. This expected behavior is illustrated in Figure 21, which shows the input noise spectrum of the MFLI LIA from Zurich Instruments. As observed, the noise decreases with frequency due to the 1/f characteristic and eventually stabilizes at a constant level. The value at which the noise floor flattens, particularly for lower amplitude settings, is used as the reference input noise level in the device's datasheet. For the MFLI, this value is approximately $2.5nV/\sqrt{Hz}$.

Figure 21 – MFLI LIA Zurich Instruments Input Noise for 3 input ranges: 100mV(red), 1mV (green) and 30nV (blue)

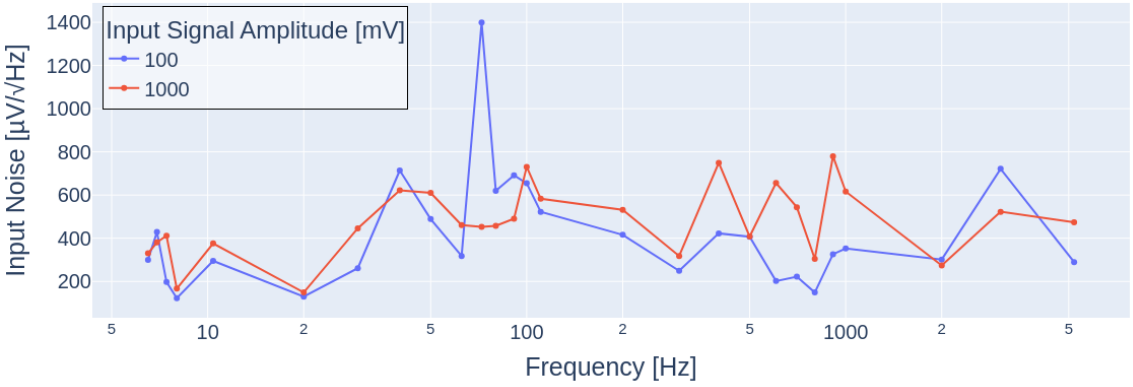


Source: (CIARDI, 2023)

To perform the input noise measurement using the custom-designed LIA, the circuit structure shown in Figure 16 was implemented on a proto-board. A $4.7k\Omega$ resistor was used as the load, and the input signal was generated and controlled via the DAC.

The resulting input noise spectrum, presented in Figure 22, deviates significantly from the expected 1/f (flicker noise) profile. Instead, the spectrum reveals additional noise components. Notably, there is no significant reduction in the input noise level with decreasing input signal amplitude, indicating that the predominant noise contributions are are not strongly related to the input amplitude. As a performance metric for the system, the mean input-referred noise was estimated to be approximately 500 $\mu V/\sqrt{\text{Hz}}$.

Figure 22 – LIA Input Noise



5.3 DYNAMIC RESERVE

The dynamic reserve quantifies the system's ability to maintain measurement accuracy in the presence of interfering signals at frequencies different from the reference. In Zurich Instruments equipment, this metric is calculated according to Equation 31, with the resulting value indicating the maximum interference level at which the measurement accuracy remains better than 1%(CIARDI, 2023). To evaluate this parameter, a National Instruments NI myDAQ device was employed alongside an adder circuit based on an operational amplifier, which was used to inject noise at a specific frequency into the system. The dynamic reserve was then determined by gradually increasing the amplitude of this interfering signal until the measurement accuracy threshold was reached.

$$DR = 20 \log_{10} \left(\frac{V_d}{V_s} \right) \tag{31}$$

Figure 23 – Module Error as a function of the Dynamic Reserve

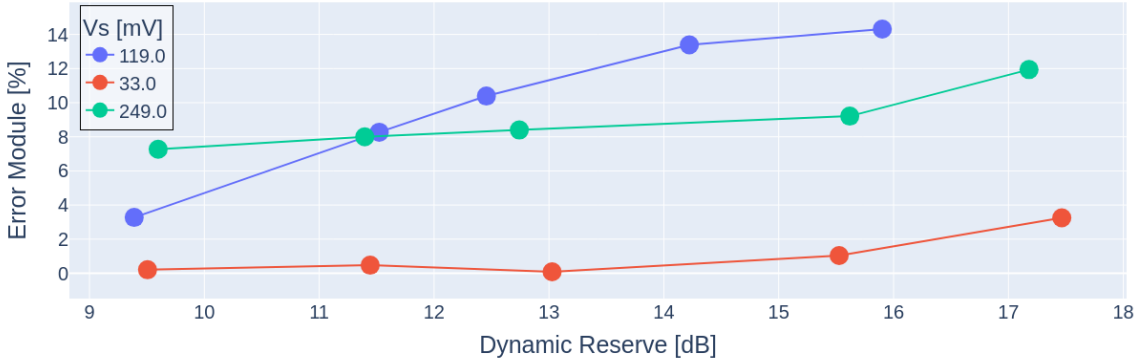
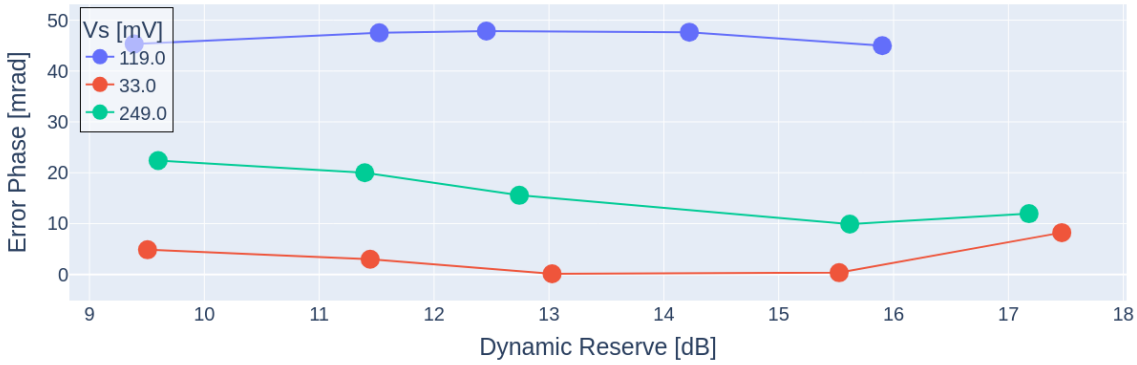


Figure 24 – Phase Error as a function of the Dynamic Reserve



To evaluate the dynamic reserve, a 100Hz noise signal was injected while the reference signal was set to 2kHz. To establish a baseline for calculating the absolute percentage error of the magnitude and the absolute phase error, a reference measurement was performed at each input amplitude (V_s) without any added noise. For each dynamic reserve value, 40,000 data points were collected, sampled at 520Hz, and the mean magnitude and phase were compared against the corresponding noise-free reference measurement.

As shown in Figure 23, the magnitude error varies with the amplitude of the desired signal, though the deviation is not directly proportional to the signal level. The phase, which ideally should remain at zero, exhibited deviations within the milliradian range—well within acceptable limits. Similarly, the phase error does not scale proportionally with the signal amplitude and remains relatively constant across different dynamic reserve values, suggesting an easier compensation.

The standard deviation of the measurements remained below 1mV, indicating that the 100Hz noise component was effectively suppressed by the low-pass filter. These low levels of variation suggest that, with proper calibration, the dynamic reserve could be further extended. The residual errors appear to be dominated by shifts in the average magnitude and phase, rather than by increased measurement noise. The valued of 14.5dB, in which the error for 33mV is smaller than 1% was considered as the performance metric.

5.4 IMPACT OF SNR ON SYSTEM PERFORMANCE

To evaluate the robustness of the measurement system under noisy conditions, this section explores how the system error varies as a function of the SNR. The aim is to assess how the accuracy of impedance measurements degrades as noise is progressively introduced into the system. By analyzing the error behavior across a range of SNR values, the system's sensitivity to noise was characterize, to identify the SNR thresholds beyond which measurement reliability significantly declines. Unlike the dynamic reserve

evaluation—where noise is injected at a single frequency—this test introduces uniform broadband noise, which affects the entire spectrum. As a result, some noise components fall within the system's passband, leading to a measurable impact on the standard deviation of both magnitude and phase.

As a baseline, impedance measurements were first conducted without any added noise. Two instruments were used for this reference: the E4980AL Precision LCR Meter from Keysight, and the develop LIA. These measurements provide a noise-free benchmark against which the system's performance under noisy conditions can be compared. The measured values of the resistive and capacitive components are summarized in Table 14.

Table 12 – Comparison between LCR Measurements and Specification for 2kHz and 2V

Component	**Specification**	**LCR**
Resistor	$1.5k\Omega$	$1.4951k\Omega$
Resistor	$2.2k\Omega$	$2.1971k\Omega$
Resistor	$3.3k\Omega$	$3.2860k\Omega$
Resistor	$4.7k\Omega$	$4.6867k\Omega$
Capacitor	$10nF$	$10.5nF$
Capacitor	$22nF$	$23.5nF$
Capacitor	$100nF$	$98.2nF$
Capacitor	$220nF$	$221.1nF$

The resistors were measured individually, and the $4.7k\Omega$ resistor was connected in parallel with each capacitor. To calculate the reference impedance, Equation 32 and Equation 33 were used. For estimating the impedance from the LIA output, the current was assumed to be constant at $2.5\text{m}A$. Based on these considerations, the results of the measurements are presented in Table 13, which includes the mean values of the impedance magnitude and phase. The standard deviations remained in the hundreds of milliohms and a few milliradians, having negligible impact on the calculated errors, then, they were omitted from the table.

$$|Z| = \frac{r_1}{1 + (\omega_1 r_1 c_1)^2} \cdot \sqrt{1 + (\omega_1 r_1 c_1)^2} \tag{32}$$

$$\theta_Z = -\tan^{-1}(\omega_1 r_1 c_1) \tag{33}$$

Table 13 – Comparison the calculated impedance and the LIA measurement

Reference		**Measurement**		**Error**	
Module [Ω]	**Phase [°]**	**Module [Ω]**	**Phase [°]**	**Module [%]**	**Phase Error [°]**
1495.1	0.000	1414.98	0.000	5.4	0.000
2197.1	0.000	2020.28	0.115	8.0	0.115
3286.0	0.000	2969.22	0.573	9.6	0.573
4686.7	0.000	4139.82	1.318	11.7	1.318
3986.10	-31.688	3538.98	-30.758	11.2	0.930
2744.78	-54.151	2591.57	-49.443	5.6	4.708
798.51	-80.170	832.34	-75.363	4.2	4.807
358.85	-85.611	365.95	-81.427	2.0	4.184

To introduce noise into the input signal, the same structure from the dynamic reserve measurement were used, with the NI myDAQ and adder circuit. The measurements were conducted at a fixed frequency of 2kHz. The noise signal was generated using a uniform distribution and injected into the system through the adder circuit. The error was calculated considering the average magnitude and phase from a reference measurement without noise insertion.

The results of these measurements are compiled in Figure 25 and Figure 26, which illustrates the system error as a function of the SNR. The figure also compares the performance of the Lock-In Amplifier at the hardware emulation stage, and the final design.

Figure 25 – Module error as a function of the SNR

Figure 26 – Phase error as a function of the SNR

The magnitude error observed in the final design is considerable, with errors exceeding 40% in the lowest SNR tested. However, as shown in Figure 27, the standard deviation remains below 3mV across all measured SNR levels. This indicates that, despite

the high error, the system demonstrates good measurement stability and can deliver accurate results for low-SNR input signals when proper calibration is applied. The magnitude error is also observed in the emulation suggesting that the deviation originates from the digital circuit, and also affects the measurement the dynamic reverse.

The phase response is notably more stable, showing smaller variations. The phase error and standard deviation remain in tens of miliradians. In comparison, the standard deviation for both the module and phase in the emulation stage is approximately three orders of magnitude lower than in the final design, reflecting the higher stability and reduced noise influence under ideal conditions.

Figure 27 – Module Standard Deviation as a function of the SNR

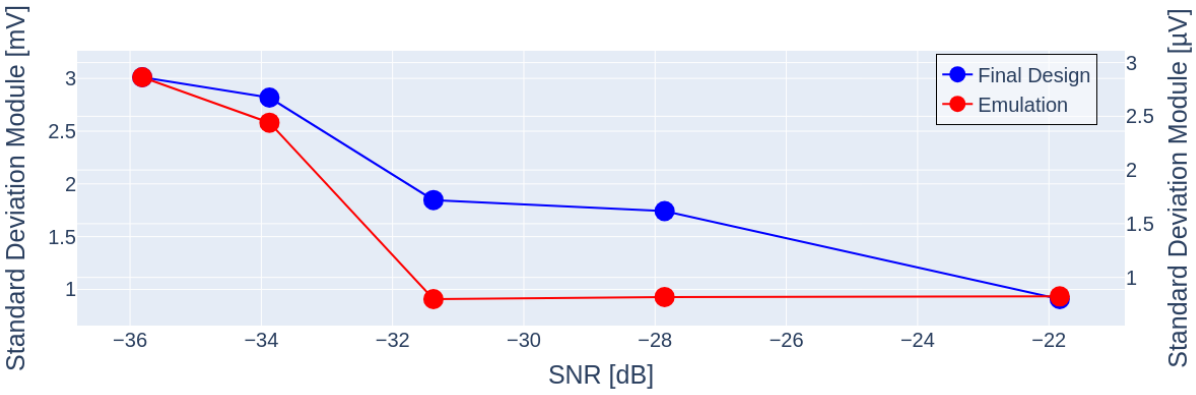
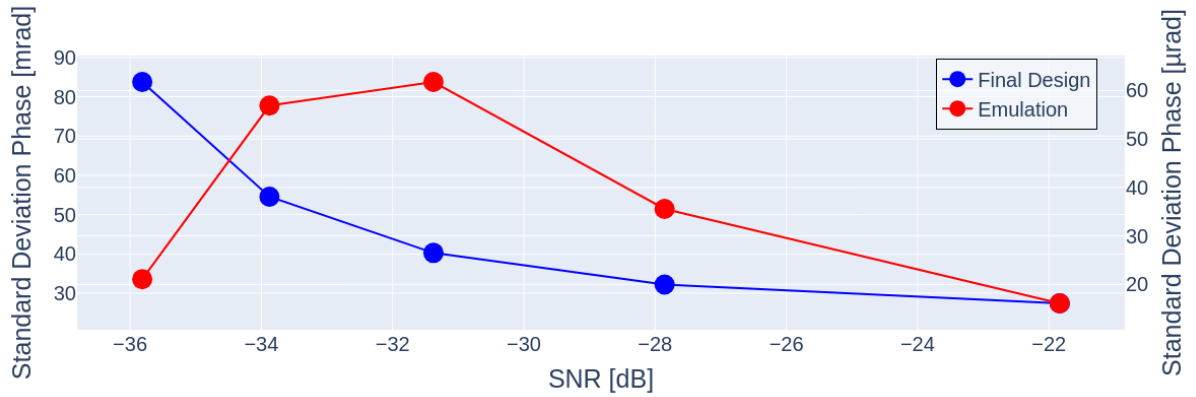


Figure 28 – Phase Standard Deviation as a function of the SNR



5.5 DESIGN CONCLUSIONS

The comparison with the MFLI from Zurich Instruments highlights that there is still considerable room for improvement in the current design. One of the most evident

advantages of the MFLI, as indicated in its specifications (ZURICH INSTRUMENTS, 2025), is its high configuration capacity. The ability to adjust the lock-in amplifier to specific application requirements is made possible through a robust and flexible control interface—something that would be beneficial to replicate in future iterations of this design. Additionally, the development of more test modules would enhance the system's performance evaluation and optimization capabilities.

Table 14 – Comparison between Zurich Instrument MFLI and the developed LIA

Specification	**MFLI**	**My LIA**
Frequency Range	0-500kHz 0-5MHz	0-26kHz
Sampling Rate	1.8ksps - 60Msps	52ksps
Vertical Resolution	16 bits	12 bits
Sensitivity	1nV	1mV
Input voltage noise	$2.5\text{nV}/\sqrt{Hz}$	$500\mu V/\sqrt{Hz}$
Dynamic Reserve	120dB	14.5dB
Demodulation Bandwidth	276μHz-205kHz	100mHz

As demonstrated in this section, the dominant source of error in the current implementation is the magnitude error, while the standard deviation of the measurements remains relatively low. This suggests that the first step toward performance improvement should be the implementation of a proper calibration procedure, enabling the LIA to adapt to varying noise levels. The calibration process is non-trivial, as the error does not exhibit a linear relationship with the input voltage level, as illustrated in the dynamic reserve analysis.

The analog front end can also be improved. The main points of attention include minimizing input noise, increasing the sample rate, and enhancing sensitivity. These specification can be improved by upgrading the DAC and ADC to higher-resolution versions and with higher communication rates.

The current source also requires improvement. Ideally, the output current should be dynamically adjusted based on the impedance under measurement to prevent saturation or overflow of the ADC input range. Additionally, the existing negative voltage supply is overly simplistic and the oscillations negatively impacts the quality of the input signal. A more robust and stable design for the negative voltage source is therefore essential. The current-source operational amplifier was not supplied with a symmetrical voltage, which may have influenced the accuracy of the results. To mitigate this issue, one possible solution is to select an operational amplifier designed to operate with a single-ended supply, or alternatively, to implement an external symmetrical power source.

Finally, there is still room to enhance the efficiency of the digital LIA implementation. In particular, optimizing the bit-width of internal operations could help reduce resource usage without compromising accuracy.

6 CONCLUSIONS

This bachelor thesis presented the design and implementation of a Digital Lock-In Amplifier based on an FPGA, supported by the introduction of essential theoretical concepts to justify design decisions. The project was developed in three main stages: a Python-based prototype, hardware emulation, and a final hardware implementation.

The first stage, implemented in Python, offered a flexible environment for testing and validating algorithmic concepts. Its ease of modification and support for fixed-point arithmetic made it a valuable tool for gaining insight into the expected system behavior before committing to hardware. The second stage focused on validating the Python results through hardware emulation and translating the logic into Verilog, serving as a sanity check for the digital design. Finally, the third stage involved performance evaluation of the complete system and the development of the analog front end, which interfaced the FPGA with external signals.

A significant portion of the effort was dedicated to mastering the Quartus toolchain for debugging and testing the digital design. Establishing reliable communication between the FPGA and the HPS also proved to be a time-consuming task, but was crucial for automating system evaluation. While the focus of this project remained on the digital aspects of the LIA, the analog front end did not receive equivalent attention. As demonstrated by the results, many of the observed limitations and error sources originate from this part of the system.

Nonetheless, this project lays a solid foundation for the development of more advanced LIA systems. The documented circuit design, the reusable Verilog modules, and the testing infrastructure provide a strong base for future work and can significantly accelerate the development process in similar projects.

One of the key focuses during development was the optimization of fixed-point operations using affine arithmetic. While this method has potential—especially in providing tight error bounds with minimal iterations—the results in this implementation were overly pessimistic. This was particularly evident when using very small filter coefficients, which led to excessive error margins.

For future developments, priority should be given to improving the analog front end to reduce noise and stabilize input conditions, including a redesign of the voltage supply circuitry. Implementing an automated calibration routine could help correct magnitude errors and better adapt the system to different noise environments. Additionally, upgrading the ADC and DAC to higher-resolution alternatives would enhance sensitivity, dynamic reserve, and accuracy. New test modules and a more configurable architecture would expand the system's usability. Finally, continued refinement of the affine arithmetic model, particularly under low-coefficient conditions, is recommended to obtain tighter, more realistic error bounds for fixed-point systems.

REFERENCES

AGUIAR SANTOS, Susana et al. System Description and First Application of an FPGA-Based Simultaneous Multi-Frequency Electrical Impedance Tomography. **Sensors**, v. 16, n. 8, 2016. ISSN 1424-8220.

AKIYAMA, Soramichi et al. Error Distribution Estimation for Fixed-point Arithmetic using Program Derivatives. In: 2024 25th International Symposium on Quality Electronic Design (ISQED). [S.l.: s.n.], 2024. P. 1–9. DOI: 10.1109/ISQED60706.2024.10528683.

ANALOG DEVICES, INC. **LTC2308: Low Noise, 500ksps, 8-Channel, 12-Bit ADC**. [S.l.], 2016. Available at: https://www.analog.com/media/en/technical-documentation/data-sheets/2308fc.pdf. Accessed: 2025-04-21.

ASLAN, Semih; MOHAMMAD, Ekram; SALAMY, Azim Hassan. Open Source Synthesis and Verification Tool for Fixed-to-Floating and Floating-to-Fixed Points Conversions. **Communications and Network**, v. 8, n. 11, p. 323–334, 2016. DOI: 10.4236/cs.2016.711323.

BOURKE, Paul. **Fourier Method of Designing Digital Filters**. [S.l.: s.n.], Mar. 1999. Available at: https://paulbourke.net/miscellaneous/filter/. Accessed: 2025-04-08.

CARMINATI, Marco et al. Compact FPGA-Based Elaboration Platform for Wide-Bandwidth Electrochemical Measurements. In: 2012 IEEE International Instrumentation and Measurement Technology Conference Proceedings. [S.l.: s.n.], 2012. P. 264–267. DOI: 10.1109/I2MTC.2012.6229507.

CIARDI, G. **Understanding the Specifications of Lock-in Amplifiers**. [S.l.: s.n.], 2023. Accessed: 2025-06-11. Available at: https://www.zhinst.com/ch/en/blogs/understanding-specifications-lock-in-amplifiers#In_noise.

DIVAKAR, Dipin et al. FPGA-Based Lock-In Amplifier for Measuring the Electrical Properties of Individual Cells. In: 2018 IEEE 13th Annual International Conference on Nano/Micro Engineered and Molecular Systems (NEMS). [S.l.: s.n.], 2018. P. 1–5. DOI: 10.1109/NEMS.2018.8556987.

ELECTRONICS, Lockdown. **Generating a Negative Voltage**. [S.l.: s.n.], 2021. Available at: https://www.youtube.com/watch?v=WaIsH6buBA8. Accessed: 2025-06-11.

FANG, C. F.; RUTENBAR, R. A.; CHEN, Tsuhan. Fast, Accurate Static Analysis for Fixed-Point Finite-Precision Effects in DSP Designs. In: ICCAD-2003. International Conference on Computer Aided Design. [S.l.: s.n.], 2003. P. 275–282. DOI: 10.1109/ICCAD.2003.159701.

GERVASONI, G. et al. A 12-Channel Dual-Lock-In Platform for Magneto-Resistive DNA Detection with ppm Resolution. In: 2014 IEEE Biomedical Circuits and Systems Conference (BioCAS). [S.l.: s.n.], 2014. P. 316–319. DOI: 10.1109/BioCAS.2014.6981726.

JIANG, G. L. et al. A New Algorithm for a High-Modulation Frequency and High-Speed Digital Lock-In Amplifier. **Measurement Science and Technology**, v. 27, n. 1, p. 015701, 2015. DOI: 10.1088/0957-0233/27/1/015701.

KISHORE, Kaushal; AKBAR, S. A. Evolution of Lock-In Amplifier as Portable Sensor Interface Platform: A Review. **IEEE Sensors Journal**, v. 20, n. 18, p. 10345–10354, 2020. DOI: 10.1109/JSEN.2020.2993309.

MACIAS-BOBADILLA, Gonzalo et al. Dual-Phase Lock-In Amplifier Based on FPGA for Low-Frequencies Experiments. **Sensors**, v. 16, n. 3, 2016. Available at: https://www.mdpi.com/1424-8220/16/3/379.

MICROCHIP TECHNOLOGY INC. **MCP4725: 12-Bit Digital-to-Analog Converter with EEPROM Memory**. [S.l.], 2021. Available at: https://ww1.microchip.com/downloads/aemDocuments/documents/MSLD/ProductDocuments/DataSheets/MCP4725-Data-Sheet-20002039E.pdf. Accessed: 2025-04-21.

PARHAMI, Behrooz. Floating-Point Representation. In: COMPUTER Arithmetic: Algorithms and Hardware Designs. 2nd. [S.l.]: Oxford University Press, 2010. chap. 17, p. 351–373. ISBN 9780195328486.

POTTS, H. **Boost your Signal-to-Noise Ratio with Lock-in Detection**. [S.l.: s.n.], 2023. Available at: https://www.zhinst.com/ch/en/blogs/boost-your-signal-noise-ratio-lock-detection. Accessed: 2025-06-24.

_____. **How Does a Zurich Instruments Lock-In Amplifier Work?** [S.l.: s.n.], 2022. Available at: https://www.zhinst.com/ch/en/blogs/how-does-zurich-instruments-lock-in-amplifier-work. Accessed: 2025-06-11.

SCIPY COMMUNITY. **SciPy v1.15.2 Manual**. [S.l.], 2024. Available at: https://docs.scipy.org/doc/scipy-1.15.3/index.html. Accessed: 2025-04-21.

SIMIĆ, Stella et al. Tight Error Analysis in Fixed-point Arithmetic. **Formal Aspects of Computing**, v. 34, n. 1, p. 3, 2022. DOI: 10.1145/3524051.

ULABY, Fawwaz T.; YAGLE, Andrew E. Filter Design, Multirate, and Correlation. In: SIGNALS and Systems: Theory and Applications. [S.l.]: Michigan Publishing, 2018. chap. 9. ISBN 9781607854912.

WOODS, Roger et al. DSP Fundamentals. In: FPGA-BASED Implementation of Signal Processing Systems. 1. ed. [S.l.]: Wiley, 2010. chap. 2, p. 11–35. ISBN 9780470030097.

YATES, Randy. **Fixed-Point Arithmetic: An Introduction**. [S.l.: s.n.], Aug. 2007. Available at:
`https://courses.cs.washington.edu/courses/cse467/08au/labs/l5/fp.pdf`.

ZURICH INSTRUMENTS. **MFLI and MFIA Specification**. [S.l.: s.n.], 2025. Available at: `https://www.zhinst.com/en/mfli-and-mfia-specification`. Accessed: 2025-06-16.