ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Implementation of a Multi-Client Network Analyzer Module for a Digital Laser Lockbox

Bachelor Thesis

Rico–Marcel Benning

January 14, 2025

Supervising Professors: Prof. Dr. Novotny, Prof. Dr. Home
Advisors: B. Dönmez, L. Milanovic

Department of Electrical Engineering & Information Technology , ETH Zürich

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Implementation of a Multi-Client Network Analyzer Module for a Digital Laser Lockbox

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Benning	Rico-Marcel

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zurich, 14.01.2024	R. M. Benning

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Abstract

The goal of this project was to develop a Network Analyzer (NA) software module based on the open-source PyRPL software for BLOOD (Bichannel Lockbox On One Device), leveraging existing FPGA hardware and software.

I successfully validated a possible NA approach and integrated the NA functionality into BLOOD's architecture. The resulting NA provides an accurate, cost-effective way to measure transfer functions of feedback loops. Future improvements include extending the NA to BLOOD's second channel, adding real-time data streaming, and refining the GUI for faster measurements.

Acknowledgment

First of all, I would like to thank my advisors Bahadır Dönmez and Luka Milanovic for their close and dedicated mentorship, advice, and for helping me gain a deeper understanding of the research group's work. I also want to thank Prof. Dr. Jonathan Home and the entire Trapped Ion Quantum Information Group for giving me the opportunity to join them and generally for welcoming me as a member of their team. Lastly, I want to thank Prof. Dr. Lukas Novotny for allowing me to write this Bachelor Thesis under his supervision.

Contents

Chapter 1

Introduction

The precision and stability of laser systems are critical in many advanced scientific applications, particularly in experiments involving trapped ions [1]. Stable lasers, locked to optical cavities, enable precise control over ion-states by ensuring that frequency, phase and intensity are staying in a desired, stable range [2]. Achieving this stability requires robust feedback systems such as Proportional-Integral-Derivative (PID) controllers to continuously adjust the laser's frequency for disturbances and deviations. In the Trapped Ion Quantum Information Group (TIQI), a custom-designed controller, BLOOD (Bichannel Lockbox On One Device), plays a central role in maintaining this stability. BLOOD's versatility allows it to function as a general-purpose controller, making it an invaluable tool for the optics laboratory [3].

In this context, the integration of a network analyzer (NA) into BLOOD represents a significant advancement [4]. Network analyzers are indispensable for measuring transfer functions of feedback loops [5]. In this project, the transfer functions of interest are those of the feedback system, specifically the response of the laser stabilization system. These transfer functions provide critical insights into the dynamics, stability, and performance of the system, allowing for precise tuning of controllers like the PI controllers used in BLOOD [3] [4]. By replacing external NAs with an integrated module, BLOOD can reduce setup complexity and enable automated, real-time diagnostics. This integration eliminates the dependency on external hardware, offering a streamlined, cost-efficient solution with enhanced functionality.

BLOOD's FPGA code is inspired by and based on a modified version of the PyRPL FPGA design. This modified code is paired with BLOOD's own software, called BLOOD Operating System (OS). The functionalities implemented in PyRPL include features such as oscilloscopes, NAs, and lock-in amplifiers. However, BLOOD OS, as its own software design, does not use all the functionalities that are included in PyRPL [3] [6].

The primary motivation for this project lies in the need to simplify and improve the feedback system's functionality. External NAs, such as the Analog Discovery 2, have traditionally been used to measure transfer functions of the laser stabilization feedback system. However, their reliance on additional hardware adds complexity and potential points of failure. Furthermore, such setups often require hardware modifications, which are time-consuming, prone to failure, and therefore undesirable in an optics laboratory environment. BLOOD's FPGA-based design offers an opportunity to integrate NA capabilities directly into BLOOD OS, using its preexisting hardware and software infrastructure. This integration not only reduces hardware dependency but also enables automated health checks and system verifications, significantly enhancing future experiments' reliability [4].

This thesis focuses on the implementation and development of a network analyzer module for BLOOD, addressing the following key aspects:

1. Exploration and validation of the preexisting PyRPL network analyzer as a potential structure for implementation.
2. Integration of the code in respect to the specific architectural nuances and requirements of BLOOD.

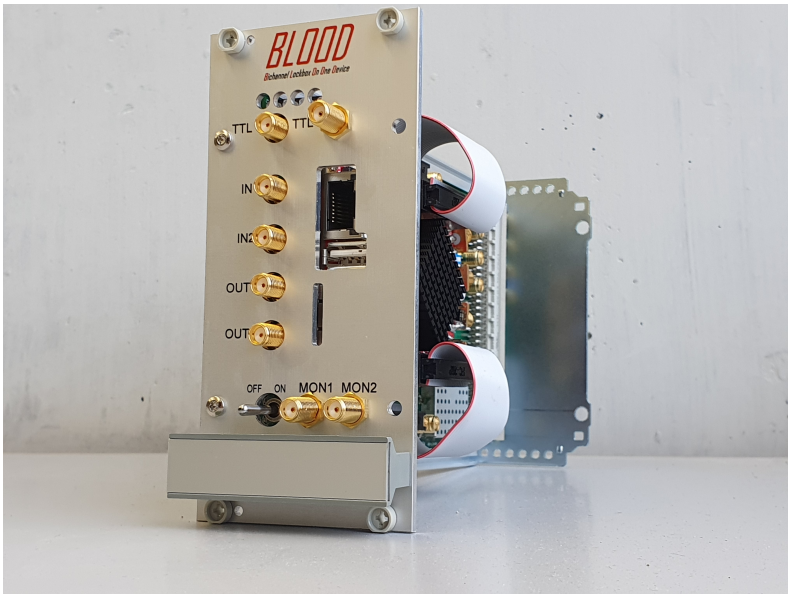


Figure 1.1: Almost assembled BLOOD, showing the front of the device. Picture taken from [3]

Chapter 2

Background

2.1 What is a Network Analyzer

The network analyzer operates by generating a sine wave and sending it through the device under test (DUT). The sine wave is used because it is a fundamental signal with a well-defined frequency, amplitude, and phase. We assume that our system is linear and time-invariant (LTI), and because of this, we know that sine functions stay sine functions. In such systems, the output at any given frequency is fully characterized by the gain and phase shift applied to the input signal. This makes sine waves ideal for frequency-domain analysis, as they enable us to isolate the system's response at specific frequencies without introducing additional harmonics or nonlinearities [2] [7].

The network analyzer performs a frequency sweep: it starts at a user-defined initial frequency and sequentially increases to a specified stop frequency. At each step, the NA compares the output signal (after passing through the DUT) to the input signal. By analyzing how the amplitude and phase of the sine wave change across frequencies, the NA generates data for constructing a Bode diagram. A Bode diagram is essentially a graphical representation of a system's frequency response. It consists of two plots:

1. Magnitude plot: Shows the gain (in decibels, dB) as a function of frequency. This illustrates how much the input signal is amplified or attenuated at each frequency.
2. Phase plot: Displays the phase shift (in degrees) as a function of frequency. This shows how the timing relationship between the input and output signals varies with frequency.

With these, in the context of control systems and electronics, one can easily assess system behavior, identify resonances, and determine stability. By

using a NA, we can accurately characterize a system's frequency-dependent behavior and therefore make it a central tool for our analysis.

2.2 BLOOD

2.2.1 History of BLOOD

In the TIQI research Group, tasks previously related to feedback laser stabilization had been initially performed with a device called EVIL (Electronically Variable Interactive Lockbox), designed by Ludwig de Clercq and Vlad Negnevitsky around 2012 [8]. This device is outdated and currently replaced by the new BLOOD (Bichannel Lockbox On One Device) that was developed during a QuanTech Workshop at ETH in 2022 [3]. BLOOD essentially implements a digital laser lockbox, supporting features ranging from arbitrary signal generation, PI controller, and IQ signal processing. Although BLOOD meets many current requirements, it was designed to be extensible so it can incorporate additional capabilities such as the network analyzer functionality.

2.2.2 BLOOD's Architecture

BLOOD is built around a Red Pitaya STEM-lab 125-14 FPGA board, which integrates a Xilinx Zynq 7010 System-on-Chip (SoC) [3]. This SoC comprises:

- **FPGA:** Handles time-critical signal processing (e.g., PI, IQ demodulation).
- **ARM CPU:** Runs a C++-based server responsible for managing communication with external clients, coordinating data exchange with the FPGA, and executing high-level software routines to enable and control BLOOD's features. This includes handling user commands, configuring parameters, and managing real-time data processing pipelines.

A custom PCB complements the Red Pitaya board to interface external signals and power the various modules, as can be seen in Figure 2.5b. On the software side, BLOOD's FPGA code was modified from PyRPL's open-source code, developed by L. Neuhaus and S. Deléglise at the Laboratoire Kastler Brossel in Paris, France. Originally created for quantum optics experiments, the package was initiated in 2014 and published under the GNU General Public License in 2017 [6]. BLOOD's lockbox architecture follows our own client-server model and code, as illustrated in Figure 2.1 [3]:

- **Python Client:** Runs on a user's PC, providing a graphical or script-based interface.
- **Server on the Red Pitaya:** Written in C++, this application bridges the client and FPGA, ensuring commands are sent to the FPGA and

real-time data is returned to the user. Here, the server also writes and reads to the registers on the FPGA.

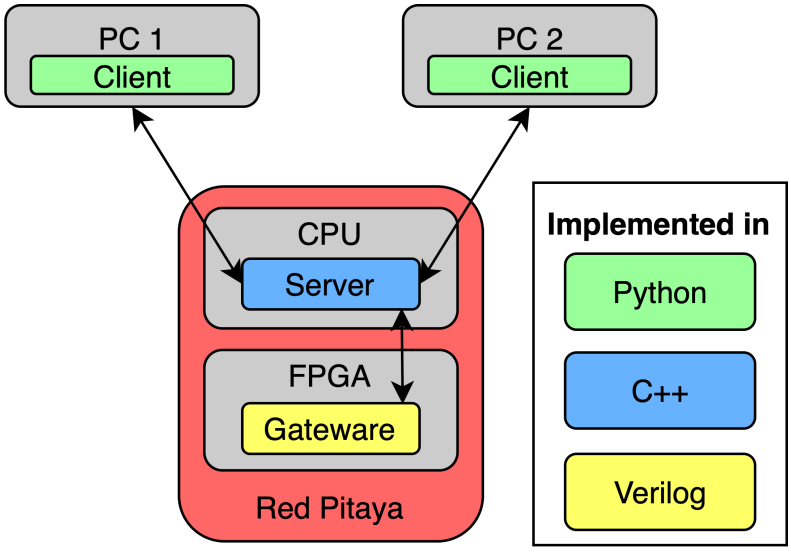


Figure 2.1: High-level abstraction of BLOOD's software-gateware stack. The diagram illustrates the communication between the clients (`Python`) on the PC, the server that runs on the CPU (`C++`), which bridges the interaction with the FPGA gateware (`Verilog`). Picture inspired from [4]

2.2.3 Multi-Client Functionality

Because multiple users may need to configure BLOOD remotely and concurrently, the server allows multi-client support. Any changes made through one client are synchronized across all clients, preventing conflicting operations. This architecture ensures that real-time signal control is handled by the FPGA, while the higher-level server application and network interface run on the ARM CPU. This separation of tasks enables fast and reliable signal processing, easy remote access, and future-proof expansion for advanced features like network analysis [3].

2.3 PyRPL's Original Network Analyzer Implementation

2.3.1 Conceptual Overview of PyRPL's Network Analyzer

PyRPL originally implements the network analyzer as a software module using the following approach:

It generates a list of frequencies based on user-defined parameters, including start frequency, stop frequency, and the number of points. After the frequency is written to the frequency register to trigger data acquisition, a stabilization

time is required. This time depends on the resolution bandwidth (RBW, which controls the smallest distinguishable frequency) and averaging settings (making data more or less precise). The quadrature data is read back from the registers and converted into amplitude and phase information. Then the results are stored in arrays for plotting or further analysis.

In essence, PyRPL's Python code iterates through each frequency step, setting the frequency, triggering or waiting for the acquisition to complete, retrieving the measured data, and moving to the next frequency. This process continues until the sweep is complete or manually stopped. All of this happens in PyRPL's client software that runs on the PC. This is distinctly different from BLOOD [9].

2.3.2 Conceptual Overview of PyRPL's Internal IQ-module

The functionality of PyRPL's network analyzer is built on fundamental signal processing techniques, including the use of In-phase and Quadrature mixers and averaging. These are essential for converting raw data into meaningful frequency-domain information. A schematic illustrating PyRPL's internal IQ-module connections is shown in Figure 2.3.

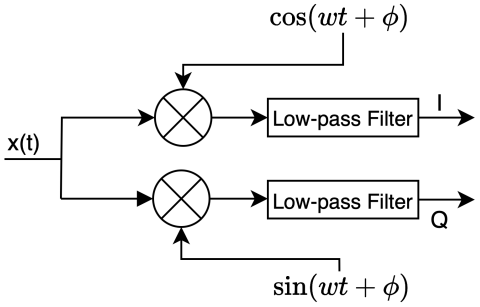


Figure 2.2: Schematic of the IQ demodulation.

IQ Mixers and Signal Demodulation

An IQ mixer is a critical component for extracting amplitude and phase information from a signal [10]. The key steps involve:

1. **Signal Mixing:** The input signal is split and mixed with two reference signals that are 90° out of phase (one cosine, one sine). This process separates the input signal into its in-phase (I) and quadrature (Q) components.
2. **Low-pass Filtering:** After mixing, the high-frequency components of the mixed signals are removed using a low-pass filter, leaving only the I and Q components at the desired baseband frequency.

3. **Amplitude and Phase Extraction:**
    - The amplitude of the signal is calculated as $\sqrt{I^2 + Q^2}$, representing the signal's magnitude.
    - The phase is derived using $\tan^{-1}(Q/I)$, capturing the phase difference between the input signal and the reference.

This approach allows the network analyzer to capture detailed frequency-response characteristics of the system. Figure 2.2 shows this in detail.

Averaging and Stabilization Time

Averaging is applied to the measured data to enhance precision and reduce noise. Key aspects include:

- Averaging Impact: By repeatedly measuring and averaging the I and Q data, random noise fluctuations are suppressed, yielding more accurate results. However, this comes at the cost of longer measurement times.
- Stabilization Time: The network analyzer introduces a short delay after setting each frequency to allow the system to stabilize. This ensures that transient effects or oscillations do not affect the measurements. The duration of this delay depends on the RBW, which determines the analyzer's overall accuracy.

Data Processing and Visualization

Once the I and Q data are captured and processed, they are converted into amplitude and phase information. This data is then stored in arrays for plotting or further analysis. The final output, often presented as Bode plots, provides insights into the magnitude and phase response of the system under test across the frequency range.

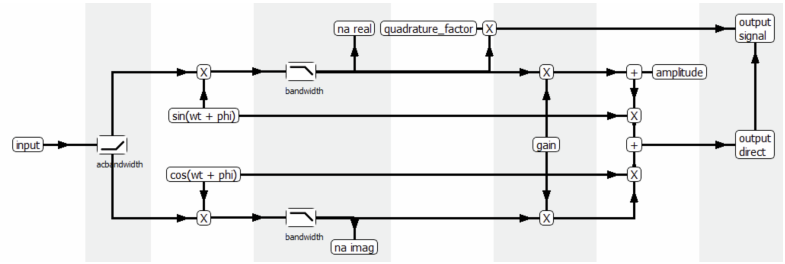


Figure 2.3: Schematic of PyRPL's internal connection of the IQ-module [6]

2.4 Some Control Theory

In the context of this thesis, it is essential to understand how a system can be characterized. First, we can differentiate between open-loop and closed-loop systems [2]:

- **Open-Loop Systems:** These systems lack a feedback path. The input directly drives the system, and any variations in the output are not corrected.
- **Closed-Loop Systems:** These systems use feedback. The output is fed back and compared to the input or a reference value. Based on this comparison, the output of the system changes to minimize the error.

In this analysis, we primarily focus on the closed-loop configuration, because open-loop systems are usually too unstable to measure practically. Especially when working with lasers, where the lock transfer function can only be accurately measured when the laser is locked. By locking the system with a controller, we ensure minimal disturbances and maintain a stable operating point throughout the measurement. We assume approximate linearity under small-signal or near-equilibrium conditions, acknowledging that the controller itself may introduce some nonlinearity. Nevertheless, this approximation holds well for the frequency ranges and signal levels of interest [2]. For simplicity, we focus only on the controller and ignore the laser. The FPGA output is looped back to the input via a cable, with the network analyzer injecting a test signal d and measuring the resulting output y. The controller output is referred to as u. The controller C, where $s = jw$ represents the complex frequency variable in the Laplace domain, keeps the system in a stable regime, allowing reliable transfer-function measurement without large deviations [2].

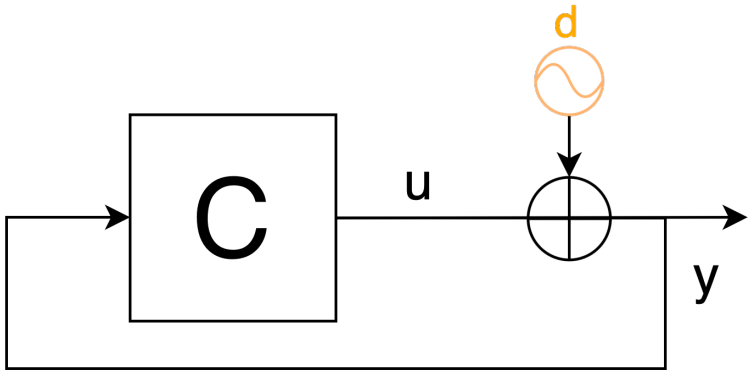


Figure 2.4: Block diagram showing the measurement configuration.

2.4.1 Derivation of closed-loop transfer function

From the block diagram in Figure 2.4, the output y is given by the relationship:

$$y = u + d \tag{2.1}$$

For the controller output we have:

$$u = C(s)y \tag{2.2}$$

Plugging Equation 2.2 into 2.1 and factoring out y, we obtain:

$$y\big(1 - C(s)\big) = d \tag{2.3}$$

The transfer function from the injected signal to the output is then:

$$\frac{y}{d} = \frac{1}{1 - C(s)} \tag{2.4}$$

Similarly, we can extract the relationship of the open-loop transfer function, expressed in respect to the measurement $\frac{y}{d}$:

$$C(s) = \frac{\frac{y}{d} - 1}{\frac{y}{d}} \tag{2.5}$$

2.5 Methodology and Implementation Overview

The development of the network analyzer module involved two key phases.

2.5.1 Verification of PyRPL's Network Analyzer

The first phase involved validating the existing network analyzer functionality within PyRPL. This was achieved by performing comparative measurements using PyRPL's network analyzer and a commercial state-of-the-art device, the Analog Discovery 2. The results were analyzed to ensure the accuracy and reliability of PyRPL in measuring transfer functions.

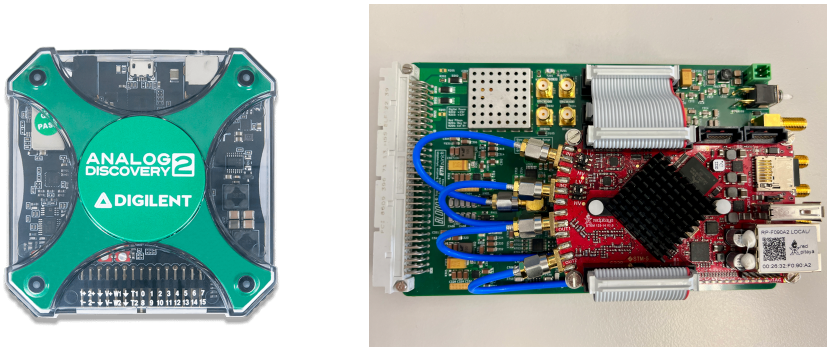
For simplicity, we consider the FPGA output connected back to the input through the cable, effectively measuring the controller itself. A network analyzer injects a test signal d, and the resulting output signal y is measured. The system is stabilized using a controller $C(s)$, as shown in Figure 2.4.

2.5.2 Integration of Network Analyzer Functionality into BLOOD

Building on the confirmed results that the PyRPL network analyzer provides reliable functionality, the second phase focused on harnessing this foundation to implement the network analyzer module directly into BLOOD's

architecture. This phase required adapting and extending BLOOD's FPGA-based design and its software stack to incorporate the network analyzer's functionalities.

These steps are detailed in the following Chapters 3.1 and 3.2.



(a) Picture of Digilent's Analog Discovery 2. Picture added from [11].

(b) Red Pitaya in connection with the PCB, from [4].

Figure 2.5: Picture of the used Oscilloscope and opened BLOOD

Chapter 3

Verification & Results

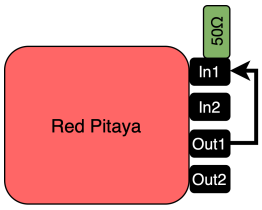
3.1 Verification of PyRPL's Network Analyzer

To validate the performance of PyRPL's network analyzer functionality, a closed-loop measurement was conducted using PyRPL on an FPGA and compared to measurements obtained with the Analog Discovery 2.

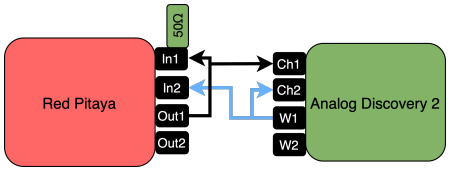
To validate the performance of PyRPL's network analyzer, we used a self-locked PI-configuration as a simple, well-understood 'toy system'. This approach ensured that we had a predictable reference response, making it ideal for verifying the NA functionality. In both setups—one using PyRPL's internal network analyzer and the other using the Analog Discovery 2—we ran the lock on the FPGA. However, when using the AD2, we needed to add the sine input on the FPGA side. To account for this, we introduced a second PI (with $P = 1$ and $I = 0$) in the FPGA. This allowed us to compare both measurement methods under nearly identical conditions. The measurement setup is displayed in Figure 3.1, as well as the internal connections of the FPGA gateware in Figure 3.2, and a schematic of PyRPL's internal connection of the IQ-module in Figure 2.3.

The results are presented in the form of Bode plots, illustrating the magnitude and phase response of the closed-loop system (Figure 3.3) and extracted open-loop transfer function (Figure 3.4). This validation was a critical step in the implementation process, as it ensured that PyRPL's existing functionality was robust. Since BLOOD's FPGA code is inspired by PyRPL software, and we planned to implement the NA based on PyRPL's NA Python code, we had to confirm the accuracy of this feature before adapting it by modifying BLOOD's C++ server code.

The Bode plots in Figure 3.3 reveal strong alignment between the PyRPL and Analog Discovery 2 measurements. Both devices exhibit consistent magnitude and phase responses across the tested frequency range, confirming



(a) Measurement Setup 1: The Red Pitaya operates in a closed-loop configuration, with Output 1 (Out1) directly connected to Input 1 (In1) with a 50Ω termination at In1.



(b) Measurement Setup 2: The Red Pitaya is additionally connected with the Analog Discovery 2 (AD2) and uses P=1, I=0. The AD2 generates a sinusoidal signal across various frequencies, feeding it into the FPGA's Input 2 (In2) and its own Channel 2 (CH2) for reference. The FPGA connects Output 1 (Out1) back to Input 1 (In1), while Output 1 is also sent to AD2 Channel 1 (CH1) for measurement. The 50Ω termination at In2 is omitted for visibility.

Figure 3.1: Conceptual Measurement Setup

the reliability of PyRPL's network analyzer functionality.

When the PI controller is configured with $P = 0$ and $I = -500$, the integral term is the only relevant factor. This single-term dominance results in high closed-loop gain near low frequencies, ensuring disturbances in this frequency range are effectively canceled. While at higher frequencies, the controller gain is low and therefore not canceling the applied disturbances, which then results in the transfer function approaching unity (0 dB). The pronounced phase lag (approximately -90°) further illustrates the integrator's influence in shaping the closed-loop response. At higher frequencies, the

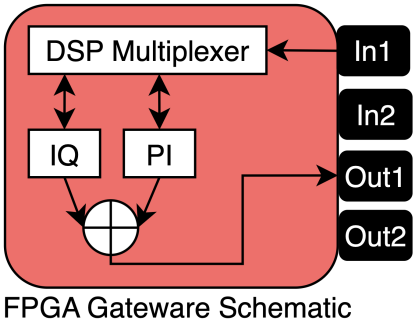


Figure 3.2: FPGA gateware schematic showcasing the internal connections of the IQ and PI modules.

integrator's amplitude contribution diminishes. Because the controller can no longer cancel those faster disturbances, the loop gain converges toward unity (0 dB), and the phase approaches 0°. Consequently, the system behaves

nearly like a direct feed-through path for higher-frequency signals.

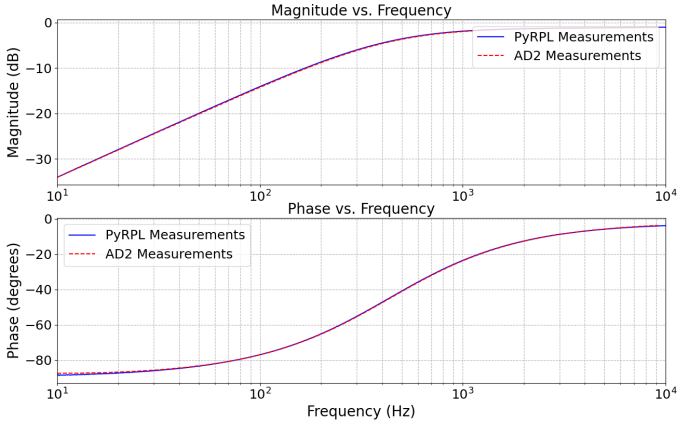


Figure 3.3: Bode plot of closed-loop measurement with given $P = 0$ and $I = -500$, showing alignment between PyRPL and Analog Discovery 2 measurements.

This integral-dominated behavior can be further verified, as discussed in Section 2.4.1, by extracting the closed-loop and open-loop transfer functions, and plotting them in Figure 3.6 and Figure 3.7. The extracted transfer function aligns closely with the Analog Discovery 2 measurements and effectively models the generic behavior of the PI controller [5]. The gain approaches infinity at 0 Hz due to the integrator's dominant influence, while at higher frequencies, the response stabilizes as the integrator's effect diminishes. This confirmation is essential as we were able to verify that the PyRPL NA performs as expected and therefore could proceed to implement it into BLOOD.

3.2 Integration of Network Analyzer Functionality into BLOOD

Following the validation of PyRPL's network analyzer, its core functionality was adapted for BLOOD. This process involved:

- **FPGA Code Integration:** Verifying the existing BLOOD bitstream to accommodate NA data acquisition and signal generation. This task was already completed by my advisor, B. Dönmez, but I still needed to review the FPGA code.
- **Server-Side Extensions:** Adjusting the C++ server-code to account for the new NA registers, ensuring accurate data acquisition, concurrent timing, and communication with multiple clients.

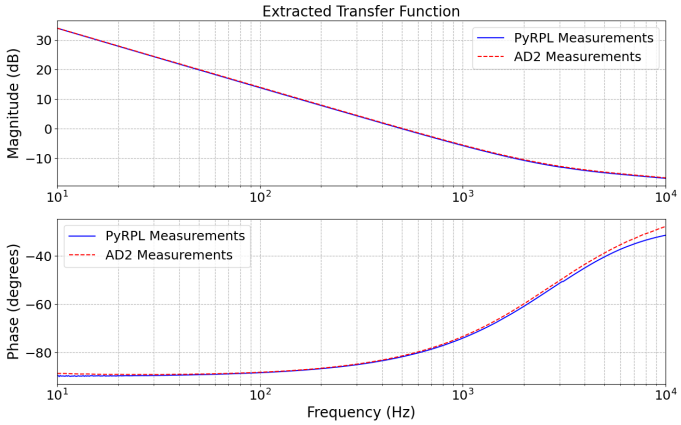


Figure 3.4: Bode plot of calculated open-loop transfer function, showing the integrator's influence.

- **Software Alignment:** Maintaining conformity with BLOOD's software architecture, notably reusing the PyRPL-derived routines for data processing, timing, and plotting, as described in Section 3.1 and illustrated in Figure 2.3.

3.2.1 Code Changes

The most significant code changes are detailed below to provide a clearer understanding of the updates and their impact, along with a reference to some recurring registers presented in Figure 3.5.

1. **Precalculation of Frequencies:**
   The process begins with the `generateFrequencies()` function, which creates a frequency vector based on the specified start and stop frequencies and the chosen plotting mode (logarithmic or linear). This vector pairs each frequency with its corresponding register values over the defined range. Precomputing these values minimizes calculation delays and prevents the need for repeated recalculations during operation.

2. **Initialization:**
   When a network analyzer sweep is initiated, the `initializeNaOn()` function sets up special registers to prevent potential bugs. This includes initializing parameters such as phase, amplitude, and channel settings to ensure proper operation.

3. **Data Retrieval and Processing:**
   The `getNaData()` function retrieves raw I and Q data from the FPGA registers, processes it, and ensures that data is read only when it is ready. This avoids errors caused by incomplete or premature data reads.

4. **Performing the Sweep:**
   The `performNaSweep()` function handles the entire sweep process. It:
   - Iterates over the precomputed frequencies.
   - Ensures the correct data is read at each step.
   - Waits for the necessary time intervals between frequency points.
   - Processes the retrieved I and Q data to extract relevant values.
   - Saves the results into a CSV file for further analysis.

Register Address	Description
`baseAddr[Dsp] + 0x20000 + 0x140`	Address to retrieve raw I and Q data from the FPGA.
`baseAddr[Dsp] + 0x20000 + 0x108`	By writing the first frequency to this register, NA acquisition is started.
`baseAddr[Dsp] + 0x20000 + 0x114`	Sets amplitude (e.g., 0.5 V).
`baseAddr[Dsp] + 0x20000 + 0x104`	Sets the phase.
`baseAddr[Dsp] + 0x20000 + 0x0`	Sets the input.
`baseAddr[Dsp] + 0x20000 + 0x4`	Turns on the output: `out1=1, out2=2, both=3`.
`baseAddr[Dsp] + 0x20000 + 0x10C`	Configures the output signal.
`baseAddr[Dsp] + 0x20000 + 0x100`	Turns on I/Q, effectively starting the network analyzer.

Figure 3.5: Some of the register addresses and their functions for the server code

3.2.2 Verification of BLOOD's Integrated Network Analyzer

To verify correct implementation, closed-loop measurements were taken using the integrated NA and compared to PyRPL's measurements (Figure 3.6

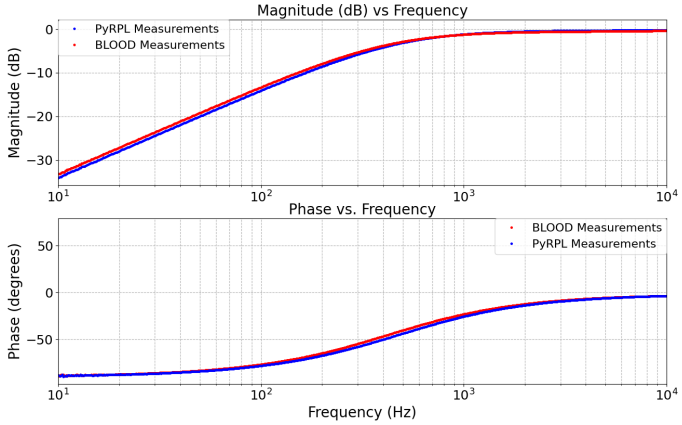


Figure 3.6: Bode plot showing the magnitude and phase response as a function of frequency for the network analyzer implemented in BLOOD compared to the PyRPL measurement.

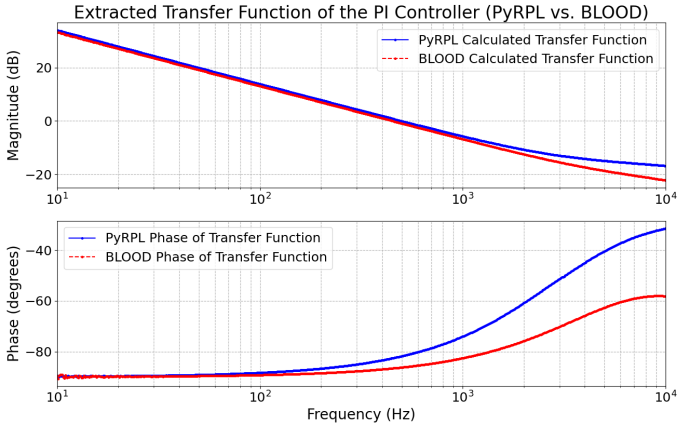


Figure 3.7: Bode plot of the calculated open-loop transfer function. Highlighting BLOOD vs. PyRPL measurements.

and Figure 3.7). The two data sets show strong alignment across magnitude and phase responses, confirming that:

- **NA Performance:** BLOOD's NA reproduces the same system behavior as PyRPL, validating its reliability.
- **Consistent Control Dynamics:** The observed integral-dominated behavior matches the theory. This illustrates an accurate integration of the NA code.
- **Minimal Differences:** Small deviations in the graphs result from slight differences in internal parameter tuning, which remain within acceptable tolerances.

Consequently, the integration of NA functionality into BLOOD eliminates the reliance on external analyzers and allows for the implementation of automatic system diagnostic tests. Furthermore, this minimizes the use of additional hardware and changes to the lab setup. As a result, real-time transfer-function measurements can now be performed directly within BLOOD's architecture, using the same user-interface flow and data-streaming architecture already established.

Chapter 4

Conclusion

4.1 Project Summary

The primary motivation for this project was to simplify and enhance feedback-based laser stabilization setups for TIQI by integrating the network analyzer functionality directly into BLOOD, thereby removing the reliance on external measurement devices. To achieve this, an existing NA approach from PyRPL was validated, adapted, and merged into BLOOD's FPGA and server infrastructure.

The results showed that BLOOD's newly integrated NA module accurately replicated transfer-function measurements of the locked system, matching the performance of both PyRPL and standalone commercial devices. This eliminates additional hardware complexity in the optics lab and provides the ability to gain knowledge about the stability of one's setup by using the NA for characterization. Furthermore, this implementation opened up the possibility for future implementation possibilities, like automated diagnostic tests using the NA.

In summary, this thesis confirmed the feasibility of implementing network analyzer functionality into the BLOOD architecture and successfully integrated core elements of this functionality.

4.2 Future Work

4.2.1 Two-channel Functionality

BLOOD features two input-output channels. While this thesis focused on the first channel for NA functionality, the same approach can be applied to the second channel.

4.2.2 Continuous Data Streaming and GUI

Currently, NA data is not continuously streamed during measurements. Future improvements include live data streaming and real-time plotting, enabling immediate feedback during measurements and variable parameter switching in the GUI.

4.2.3 Timing Optimization

Measurement points use fixed intervals, which can slow down data collection at higher resolutions. Introducing variable timing intervals would reduce waiting periods and accelerate measurements.

4.2.4 Internal Parameter Optimization

Minor variations in transfer functions indicate the potential for parameter fine-tuning. Further optimization would help mitigate these small deviations and allow for more accurate measurements.

Bibliography

[1] H. HAFFNER, C. ROOS, and R. BLATT, "Quantum computing with trapped ions," *Physics Reports*, vol. 469, no. 4, pp. 155–203, Dec. 2008, ISSN: 0370-1573. DOI: 10.1016/j.physrep.2008.09.003. [Online]. Available: http://dx.doi.org/10.1016/j.physrep.2008.09.003.

[2] R. Oswald, "Characterization and control of a cryogenic ion trap apparatus and laser systems for quantum computing," Doctoral Thesis, ETH Zurich, 2022. DOI: 10.3929/ethz-b-000603723.

[3] G. Bisson, M. Glantschnig, D. Hagmann, and P. Tirler, *Network-enabled digital lock box for trapped-ion experiments*, May 2022. [Online]. Available: https://tiqi.ethz.ch.

[4] D. Panfilova, *Implementation of anti-windup and characterization of the blood device for laser locking*, Sep. 2024. [Online]. Available: https://tiqi.ethz.ch.

[5] L. Milanovic, "Enhancement of fabry-pérot cavity-locks using fpga-based filters," M.S. thesis, ETH Zurich, 2024. DOI: 10.3929/ethz-b-000647575. [Online]. Available: https://doi.org/10.3929/ethz-b-000647575.

[6] L. Neuhaus and S. Deléglise, *Pyrpl's website*. [Online]. Available: https://pyrpl.readthedocs.io/en/latest/ (visited on 01/08/2025).

[7] J. Verspecht, "Large-signal network analysis," *IEEE Microwave Magazine*, vol. 6, no. 4, pp. 82–92, Dec. 2005. [Online]. Available: http://www.janverspecht.com (visited on 12/27/2024).

[8] L. de Clercq, "Transport quantum logic gates for trapped ions," Ph.D. dissertation, ETH Zurich, 2015. DOI: 10.3929/ethz-a-010584586. [Online]. Available: https://tiqi.ethz.ch/publications-and-awards/phd-theses.html#y-2015.

[9] L. Neuhaus and S. Deléglise, *PyRPL: Network Analyzer Module*. [Online]. Available: https://github.com/pyrpl-fpga/pyrpl/blob/main/pyrpl/software_modules/network_analyzer.py (visited on 12/27/2024).

[10] M. Microwave, *The why and when of iq mixers for beginners*, Jun. 2015. [Online]. Available: https://markimicrowave.com/technical-resources/application-notes/the-why-and-when-of-iq-mixers-for-beginners/ (visited on 12/27/2024).

[11] D. Inc., *Analog discovery 2: 100ms/s usb oscilloscope, logic analyzer, and variable power supply*. [Online]. Available: https://digilent.com/shop/analog-discovery-2-100ms-s-usb-oscilloscope-logic-analyzer-and-variable-power-supply/?srsltid=AfmBOorP46uY_EzpL2CSe3EzXCRzc5f0V1YJKuSkJ3Ka27PHXV-Gr0Ph (visited on 12/20/2024).

Appendix A

Declaration of AI Usage

I hereby declare that I used AI in this thesis solely to correct word and grammar errors and improve overall vocabulary. The ideas, concepts, and content represented are entirely mine. The models I used were ChatGPT-4 and o1, accessible at https://openai.com.