An Investigation of USRP FPGA as a platform for Quantum Sensing and Control

by
Aaron Wubshet
B.S., EE MIT (2019)

Submitted to the Department of Electrical Engineering and Computer Science
In partial fulfillment of the requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
May 2020

An Investigation of USRP FPGA as a platform for Quantum Sensing and Control

by

Aaron Wubshet

B.S., EE MIT (2019)

Submitted to the Department of Electrical Engineering and Computer Science

In partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

Abstract

This work develops a digital feedback control system for a quantum system in an effort to determine the viability of the Universal Software Radio Peripheral (USRP) 2954R for quantum sensing and control. Conducted alongside Lisa Poyneer and John Breneman at the Lawrence Livermore National Laboratory (LLNL), this work evaluates the radio frequency and signal processing capabilities of the USRP platform in relation to an array of other platforms, develops workflows for incorporating existing algorithms into the USRP environment and ensuring the portability of the algorithms to other Field Programmable Gated Array (FPGA) platforms. This work confirms the USRP as a viable platform for quantum sensing through experimentation on the quality of the radio frequency front end and USRP development environment.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. IM RELEASE NUMBER LLNL-TH-809963.

Thesis Supervisor: Jacob White
Title: Professor of Electrical Engineering and Computer Science
Thesis Supervisor: Joseph Steinmeyer
Title: Principal Lecturer of Electrical Engineering and Computer Science
Thesis Supervisor: Lisa Poyneer
Title: Project Principal Investigator at Lawrence Livermore National Laboratory
Thesis Supervisor: John Breneman
Title: Group Leader at Lawrence Livermore National Laboratory

Acknowledgments

Thank you to everyone that helped make this work possible. From the engineers and physicists at LLNL to my friends and professors at MIT, I couldn't have completed this thesis without immense help. I want to acknowledge my mentors and supervisors at LLNL, John Breneman and Lisa Poyneer, for providing constant support on all aspects, both technical and not. I also want to thank Jacob White and Joseph Steinmeyer for supervising my work from over 3,000 miles away giving the freedom to explore and conduct pure research. While being far away from home in Livermore, CA was a new experience, I want to thank my host Cindy for making her home feel like my own. Finally, I want to thank my mom for pushing me to be the best version of myself from day 1.

Contents

List of Figures

Chapter 1

Introduction

Futuristic computing conjures up images of flying cars, holograms, and quantum computers capable of computing the solution to any problem imaginable. While hovering vehicles [18] and hologram-like projection technology [17] are here and being developed rapidly, scalable quantum computing remains a “10 year technology", forever 10 years away. Quantum computing suffers from an imbalance of hardware progress with software progress. Functional hardware struggles to keep up with the rapid development of theoretical quantum algorithms. This work aims to take a step toward remedying that by validating the Universal Software Radio Peripheral (USRP), an FPGA platform, for estimating and controlling quantum states.

1.1 Motivation

Quantum systems in general, not just quantum computers, provide a door into a wide range of new advancements in science and technology. The term quantum systems here is an umbrella term for many different physical systems with many possible implementations. Two promising implementations are superconducting qubits and ion traps. The former involves reducing the temperature to near absolute zero in a dilution refrigerator and interacting with the qubit via RF signals [7]. The latter stores the qubits as ions trapped in free space using electromagnetic fields [9]. While each system faces a unique set of challenges and thus requires a unique approach, this work attempts to demonstrate that tracking and controlling quantum states is plausible and provides ubiquitous benefits. Also agnostic of the system approach, there are a wide range of applications including encryption and axion detection research. The former involves making advancements in hardware to factor large numbers while the latter hopes to leverage the sensitivity of quantum states to make difficult measurements

Modern day encryption algorithms depend heavily on the lack of an algorithm that can factor large numbers in reasonable amounts of time. However, quantum algorithms exist that are the exception to this trend. One such algorithm, Shor's algorithm, factors numbers quickly with a runtime that is polynomial in $log(N)$, where N is the number of bits required to represent the number being factored. Shor's algorithm was purely theoretical until researchers were able to realize the algorithm on hardware, first at IBM using an NMR implementation [19] and then more recently at the University of Bristol using photonic qubits [11]. While neither of the numbers that were factored are very large (21 for the photonic approach and 15 for the NMR approach), a major hurdle in factoring larger numbers is the requirement of larger quantum systems that actually behave according to the Hamiltonians used to describe them. This requires hardware that can accurately measure and control the state evolution of the quantum system in question. The notion of "quantum control" in this context has two parts. The first refers to the ability to control the evolution of the state or states of a quantum system at the appropriate time scale, and the second is doing so while avoiding decoherence, the collapse of the quantum state to a classical one [1]. It is worth noting that both implementations of Shor's algorithm are different from the two leading quantum computer implementations discussed before, superconducting qubits and ion traps. Superconducting qubits, the focus of this work, provide the benefit of scalability and this work aims to demonstrate a platform that has the potential to increase coherence times.

In contrast to the practical appeal of quantum systems for furthering encryption research and applications, axion detection offers a different allure. The axion is a theoretical particle believed to be the silver bullet for physicists providing a solution to two important problems: what is dark matter [13] and why does quantum chromodynamics seem to preserve CP-symmetry [12]? The theories surrounding the answer to the latter rely upon the existence of axions. At the same time, axions are a likely candidate for the answer to the former showing promise as a possible subset of cold dark matter. However, the experimental search for axions, suffers from the perennial issue of noise obstructing measurements. Thus, physicists interested in axions have become interested in the possibility of using quantum systems as part of their experimental apparatuses. Specifically, utilizing Superconducting Quantum Interference Devices (SQUIDs), experiments are being run using a microwave cavity and extremely strong magnetic fields in hopes of detecting axions [15]. The experimental hypothesis revolves around axions disturbing the quantum states, so if the quantum state evolution can be measured or estimated and subsequently compared against the theoretical evolution, a fast reset signal can drive the quantum system to a known state and watch the state evolve in the presence of some unknown particle, believed to be an axion. The particular interest in a work like this one, lies with the fact that the platform being used is an FPGA and thus can provide that reset signal on the time scale of the evolution of the quantum system.

These applications demonstrate a need for progress in hardware for quantum systems. This thesis contributes to that goal by focusing on the implementation of an optimal control algorithm on the USRP along side a host side data viewer. Previous work (section 1.3) has demonstrated analog control of transmons (section 1.2.1), FPGA based digital feedback control of transmons, and USRP manipulation of quantum gates. The encompassing project that this thesis fits within draws from these 3 key achievements in an effort to demonstrate the USRP as a viable FPGA platform for feedback control of transmon state. By developing and implementing a sophisticated, high speed, and compact control algorithm that is both optimal but can still operate on an FPGA, this work contributes to the goal of quantum hardware advancement.

1.2 Background

In order to continue the discussion, a brief review of the core concepts employed throughout this work is necessary. The work was conducted by engineers, so the core of the work is to solve the engineering question of if the USRP is viable platform for quantum control. Thus, FPGA and controls concepts are utilized heavily. However, the experimental apparatus was created and maintained by physicists, so a bit of quantum physics knowledge will be helpful during the discussion.

1.2.1 Physics

The fundamental building block for quantum systems is the qubit. Analogous to traditional bits in classical computing, qubits are the basic unit of information. Classical computing usually uses complementary metal oxide semiconductor (CMOS) transistors to physically manifest the voltages that represent bits, high voltage for a 1 and low voltage for a 0. The quantum analog, qubits, are dependent upon the notion of superposition which allows the qubit to have a probability distribution across its possible states leading to the popular phrase 'the qubit is 0 and 1 at the same time.' The implementation of qubits for quantum systems is not yet standardized, but superconducting charge qubits offer promise for their scalability and longer coherence times. In order to create and maintain the superconducting charge qubits for the system used in this work, extremely low temperatures are required. At near absolute zero, the superconductive properties manifest. This work utilized a charge qubit circuit architecture known as a planar transmon. A transmon is a non-linear, superconducting circuit that utilizes a Josephson Junction (JJ). Typically a JJ is a thin (micron scale) layer of non-superconducting material sandwiched between two superconducting layers. In a transmon, the two layers are capacitively shunted together to minimize charge noise. It can be configured to behave as a "qubit" [2] as in this work which utilized a planar structure to increase coherence times to as high as 40 us. In order to actually utilize the transmon, a method of measurement is necessary. Thus, the transmon is placed inside of an aluminum cavity, sized to allow for

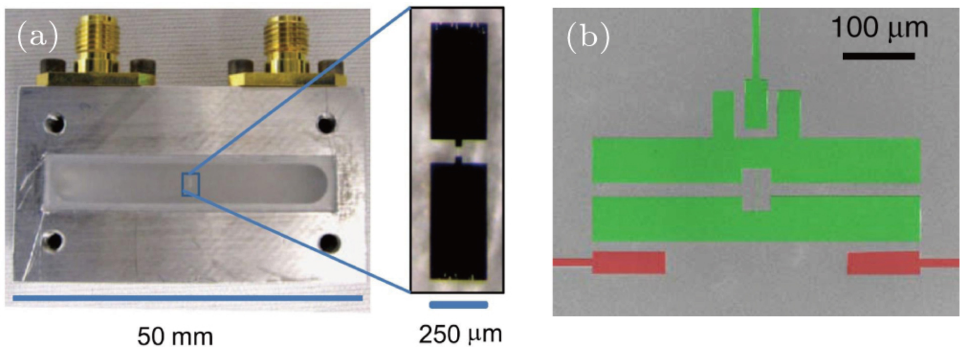


Figure 1-1: The figure above shows the physical structure of the cavity with the transmon inside. The physical size constrains the frequency used to couple with the cavity and "measure" the transmon. This image is courtesy of Liu Wei-Yang.

microwave RF tones to couple to the cavity. Thus, the cavity has a fundamental frequency in addition to the qubit itself having a frequency that can be used to extract information by supplying an input signal tuned to the qubit to prepare it and then a second signal tuned to the cavity frequency and comparing the response signal to the initial cavity signal. Exciting the system at the qubit's resonant frequency can be used to change the actual state of the qubit (including putting the qubit in a superposition) while using the cavity's resonant frequency will produce information about the state. Figure 1-1 shows an example of a cavity with the transmon inside. There are a variety of signal pulse shapes and power levels that can be leveraged to extract different types and amount of information from the cavity plus qubit system. This work is mainly concerned with what is known as a weak measurement that disturbs the quantum system very little but also provides very little information. The information manifests as a phase shift between the input signal tuned to the cavity frequency after it has gone through the quantum system relative to the signal before it went through the quantum system. The greater the strength of the measurement the greater the measurement "back action" which results in quicker decoherence. On the flip side, the weaker the measurement, the greater the relative role noise plays in the measurement. By providing a resonant input, or Rabi oscillation drive, to the quantum system, the system begins to oscillate but quickly dephases and decays due to the quantum measurement back action of a weak measurement. This *dephasing* is the crux of the experiment [5]. This dephasing occurs in this systems with a time constants of less than 10 us, despite achievable coherence times of up to 40 us.

1.2.2 FPGA

Implementing any algorithm or application on an FPGA presents a set of challenges unique from typical CPU programming, an issue that will be discussed in more depth in Chapter 5. However, there are a few key architectural points worth mentioning now. The first is pipelining. Figure 1-2 gives a brief overview of pipelining. Traditional processors come prebuilt with a multi-stage pipeline, but when implementing a new algorithm or application on an FPGA, the task of pipelining is left to the application developer.

It is unavoidable in a situation such as this work because of the extremely small time scale at which the algorithm must process data and provide results in order to stimulate the system before it has evolved too far. However, the additional stages create an initial delay in the output. This coupled with the fact that the algorithm of this work operates in a burst, not continuous, fashion means that the initial delay actually is relived periodically whenever the reset is pulsed. However, as long as the delay is small compared to the time scale of the quantum system's evolution, it is manageable. This timing constraint is further impacted by the precision used at any point in the data flow. Thus, fixed point values with precision limited to what was exactly needed were used at all times. This structure of the FPGA implementation leads to a stark tradeoff in performance and compile time. As the algorithm was built up, compile times grew from minutes to hours to days, a common hallmark of FPGA programming. The compilation process can be divided into the synthesis stage and the place and route stage. The synthesis simply checks the validity of the FPGA code (VHDL in the case of this work). The place and route sections actually work on determining a valid organization of the logic blocks that can successfully implement the algorithm created by the VHDL. This place and route stage also determines whether the desired clock rate is achievable in hardware. Often times, a majority of the compilation time is spent running different optimization algorithms in order to find a layout and routing scheme that meets timing.

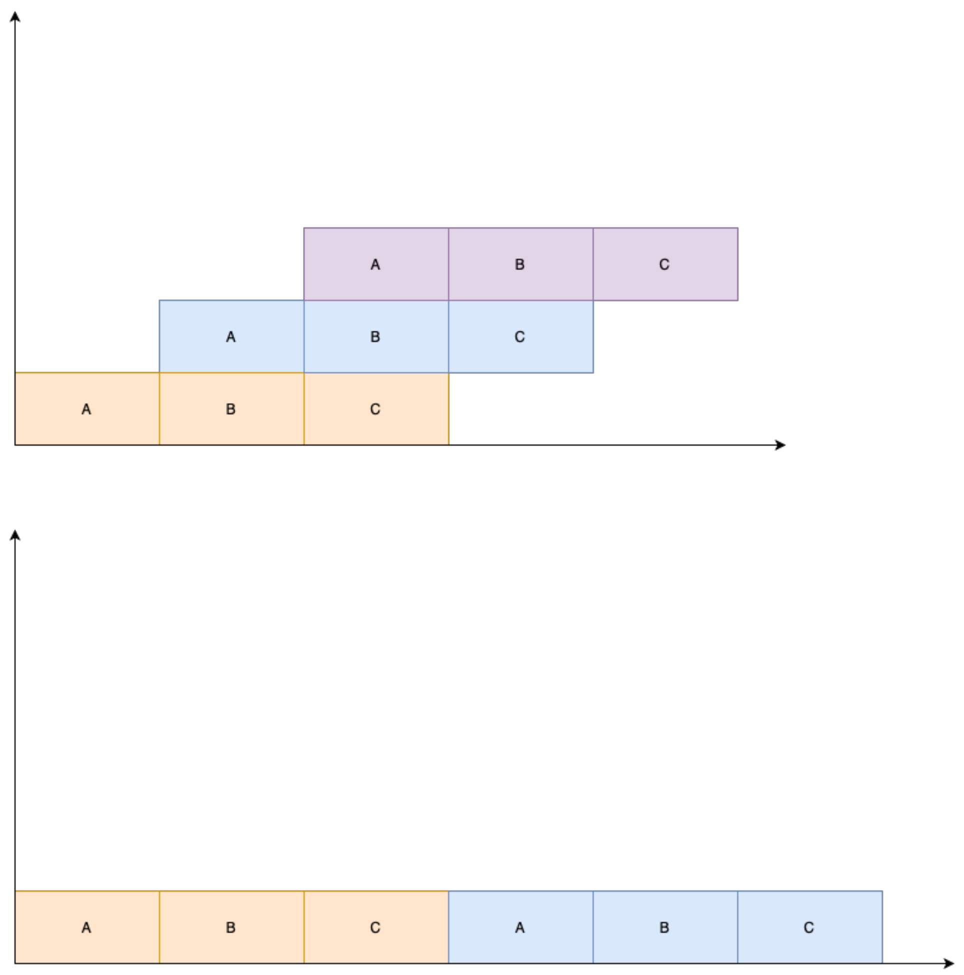


Figure 1-2: This diagram provides a visual explanation of how pipelining works. Each color corresponds to a single operation or task. Each letter corresponds to the modular basic functions that the processor can implement. Thus, increasing throughput is a matter of ensuring that as many of the different modular functions are being utilized at the same time as possible.

1.2.3 Controls

At the end of the day, the goal is to actually *control* the quantum system, and thus, an understanding of the underlying control concepts is critical. The first notable point is this control system is constructed as a discrete time controller. Each time step samples a new weak measurement and generates a new output command signal, with some allowances for initial pipeline delay (see section 1.2.2) and windowing effects which will be discussed in Chapter 3 as part of the algorithm. Figure 1-3 gives a side by side of a generic feedback system and the analogous feedback structure used in this work. The algorithms implemented on the FPGA are analogous to various portions of the feedback loop. The quantum system in the fridge can be represented as the plant block and serves as the sink for any control system command signals and the source of any system output signals. The sensor block maps to the initial steps of the control algorithm that measures the dephasing of the system. The estimator block maps to the estimation portion of the quantum control algorithm. Finally, the controller block actually maps to the final command signal calculation portion of the control algorithm by taking a desired or reference signal in combination with the estimation of quantum state evolution (based on the dephasing of the system) to generate the appropriate stimulus for the quantum system plant to extend the quantum coherence. This basic control theory setup with a plant, sensor, estimator, and controller utilizes a modular structure which if adhered to in the actual implementation allows for more sophisticated versions of each block to be easily substituted resulting in a more robust implementation. For this work, a relatively high fidelity sensor was used along with a sophisticated estimator while the controller studied (pseudo-PLL) was relatively simple.

1.3 Prior Work

Previous work at the Delft University of Technology [14], UC Berkeley [20], and Yale [10] has demonstrated the ability to measure and control one or two transmons. These achievements were part of the inspiration for this work. The Berkley group, for example, had a quantum system and analog hardware that measured the dephasing and corrected for it using an analog phase-locked loop. The dephasing results in decaying Rabi oscillations, but with a feedback control system, the Berkeley team was able to stabilize the Rabi oscillations indefinitely and generated Figure 1-4. The Delft UT work employed a similar approach but used a predetermined threshold rather than a PLL. One of the key facts here is that both systems used analog hardware which has limitations in the sophistication of the controller that can be implemented. The Yale work provides yet another indication that this current work has promise

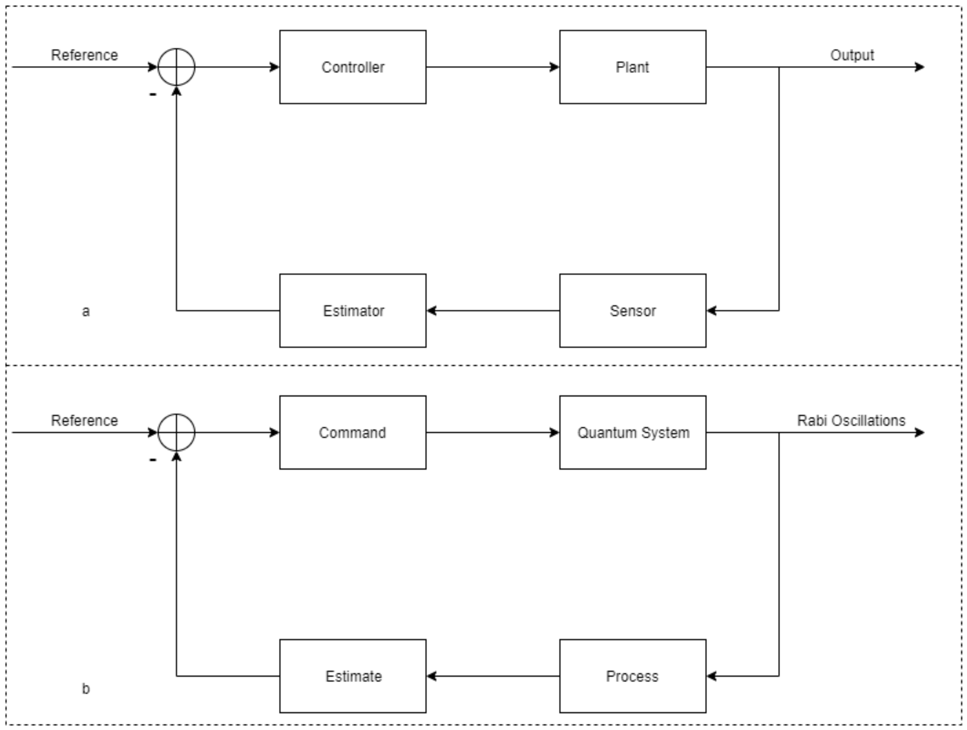


Figure 1-3: This diagram draws an analogy from the generic plant/controller/sensor/estimator paradaigm to the portions of the QSCA arranged in a simple feedback loop.

by demonstrating that an FPGA can be used to do digital feedback control with simple control schemes. Another large scale collaboration [4] actually was able to demonstrate quantum gate control using the USRP which furthered the hypothesis that the USRP could serve as a viable hardware platform. Ultimately, this work is concerned with the possibility of using digital hardware because of the advantage of implementing optimal control strategies. The specific type of digital hardware selected was an FPGA (a comparison of other digital hardware options are discussed in Chapter 5) for the flexibility and performance specifications.

1.4 Current Work

Thus, a major driver for this experiment is to demonstrate that with sufficient digital hardware, a more sophisticated control scheme can be implemented. The proposed FPGA platform is the USRP. It is a software defined radio with an FPGA for control

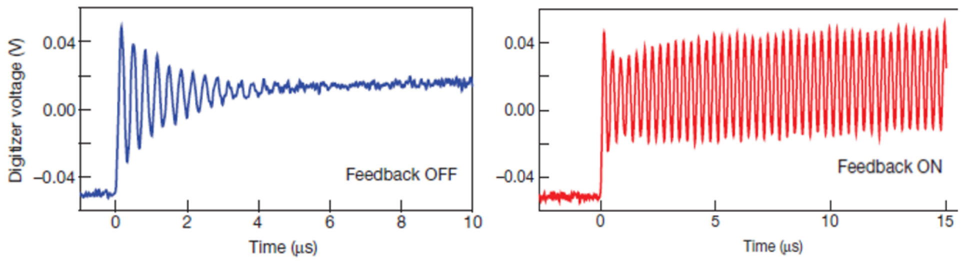


Figure 1-4: This image demonstrates what success looks like by sustaining Rabi oscillations using feedback. This result was achieved using analog hardware at Berkeley. The image is courtesy of Vijay et al

and data processing that has shown promise for wireless communication research [6] at similar frequencies to what this work operates near.

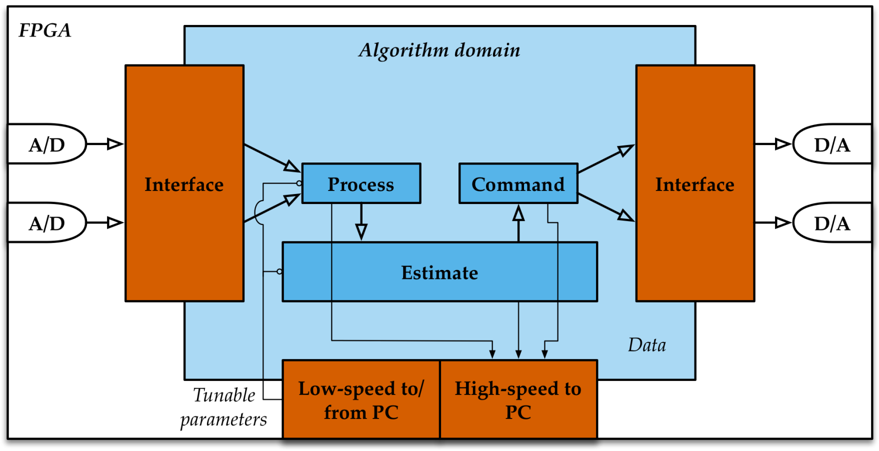


Figure 1-5: This diagram lays out the important interfaces and algorithm segments explored in this work. The ADCs take inputs from the quantum system and interface with the FPGA (details of which depend on the platform). The data is then pipelined through a 3 stage algorithm (process, estimate, and command) before interfacing with the outside world via data sent back to the user/host and stimuli generated via the DACs. This image is courtesy of Lisa Poyneer.

Figure 1-5 shows the overall layout of the processing. The inputs from the experiment come into the USRP through ADCs and interact with a device specific interface before going through the FPGA processing. This is also demonstrates the need for a user interface that sends data back to the host. Afterwards, the output from the FPGA goes through the DACs in order to actually drive the quantum system in feedback. The focus of this work will be on creating the appropriate interface implementation to support the existing FPGA processing as well as build out more of the FPGA processing. The FPGA processing, which will be discussed in more detail in Chapter 3, can be broken into 3 steps: Process, Estimate, and Command. The Process step is effectively a phase difference measurement between the two RF tones (one that has interacted with the system and one that hasn't), signal and reference. The Estimate step implements an algorithm to determine how the probability of the transmon's state will evolve. The Command step applies a varying amplitude signal to appropriately modulate the output RF tone used for control. The final step, Command, was built out as part of this work while the first two steps were for the most part completed and in the testing phase for the duration of this work. The main addition to the project brought by this work was the USRP. None of the processing had been targeted to the USRP until this work, though for a significant portion of time, the Innovative Integration (II) X6-1000M FPGA platform was used. In an effort to modernize and simplify the system, this study into the viability of the USRP platform began.

Chapter 2

Experimental Process

Ultimately, this thesis work involved a combination of software development and hardware interfacing. The embedded application development and data analysis demanded a wide range software tools, and the exotic constraints of superconducting qubits required a lot of specialized hardware. Luckily, the laboratory experimental setup, which will be described in this chapter, was mostly predefined removing the uncertainty of various quantum systems and how they behave. It also provided a fixed hardware interface to develop toward on the USRP.

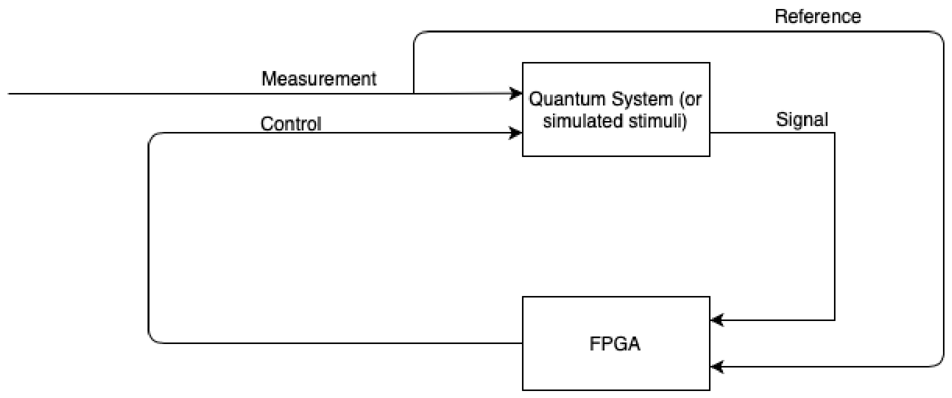


Figure 2-1: The diagram above gives the high level layout of the experimental apparatus. The measurement and control tones are inputs to the quantum system while the signal and reference tones provide inputs to the FPGA.

2.1 Physical Apparatus

In order to validate the USRP as a quantum control platform, it was inserted into a feedback loop with a quantum system. It served the role of sensor, estimator, and controller in a classical control context. The "plant" it interacted with was a quantum system or simulated stimuli depending on the stage of testing. A block diagram of the high level organization of the experiment is illustrated by Figure 2-1. The quantum system is excited by a Measurement and Control RF tone and responds with a RF tone that this work will refer to as "signal" while referring to the Measurement RF tone as "reference" before entering the FPGA for processing. The Control RF tone is then produced by the FPGA to correct the quantum system. The details of the experimental setup depends on whether the experiment is conducted in the laboratory or in a bench top fashion. The former involves an actual quantum system inside a dilution refrigerator while the latter simulates the output of the quantum system using arbitrary waveform generators (AWGs).

2.1.1 Bench top Testing

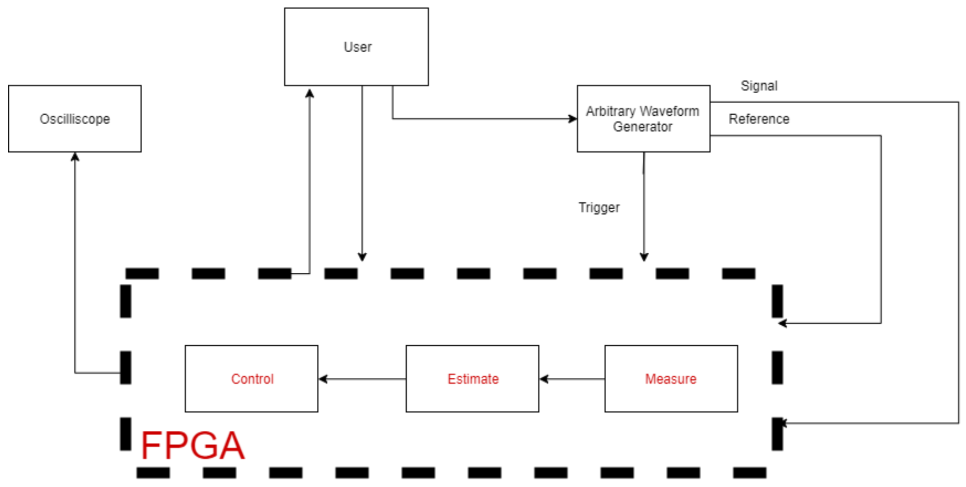


Figure 2-2: This diagram specifies the layout of the bench top testing phase in which the algorithm running on the FPGA (USRP or otherwise) is stimulated using an AWG and data is sent back to the host and viewed on an oscilloscope to ensure that known inputs resulted in the correct outputs.

Bench top experimentation is a crucial step to verify functionality before beginning laboratory experiments. Figure 2-2 describes how the USRP interacts with simulated quantum system output. In this scenario the AWG simply generates a RF tone which the USRP can internally mix down to the desired intermediate frequency (IF) of 25 MHz and then process as inputs to the algorithm that does the sensing, estimating, and controlling. Chapter 3 will discuss the details of the Quantum Sensing and Control Algorithm (QSCA). The AWG also provides the trigger for data collection on the USRP. While most of the connections to the USRP are SMA cables, the trigger must be routed through a D-subminiature 15 breakout board. The data collected is sent through the QSCA before being sent back to the user for display and saving to disk (and later analysis and post processing) over a PCIe x4 channel cable for high throughput data.

2.1.2 Laboratory Testing

When laboratory experiments are conducted, the quantum system being used is a transmon held inside a custom aluminum cavity kept at 7 mK inside of a Bluefors dilution refrigerator named Jarvis. Figure 2-3 outlines the experimental apparatus when in the lab.

The qubit's ground to first-excited state transition frequency is around 4.5 GHz which corresponds to the Control RF label entering the quantum system. The mixer is an optional element depending on the frequency capabilities of the FPGA in use. The cavity's fundamental frequency is around 7.5GHz which corresponds to the Measure RF label entering the quantum system in addition to being routed around as the reference line. Both the transmon and resonator are addressed by microwave pulses that couple to the cavity. Microwave signals leaving the cavity are amplified by a Josephson traveling wave parametric amplifier (TWPA) located at base temperature, a 4 K high-electron-mobility transistor (HEMPT) amplifier, and room temperature amplifier. The response from the transmon in the fridge is amplified and mixed to 25 MHz. These two signals, signal and reference are the fundamental inputs to be processed on an FPGA.

After the FPGA has completed its processing, data is pushed out to users in

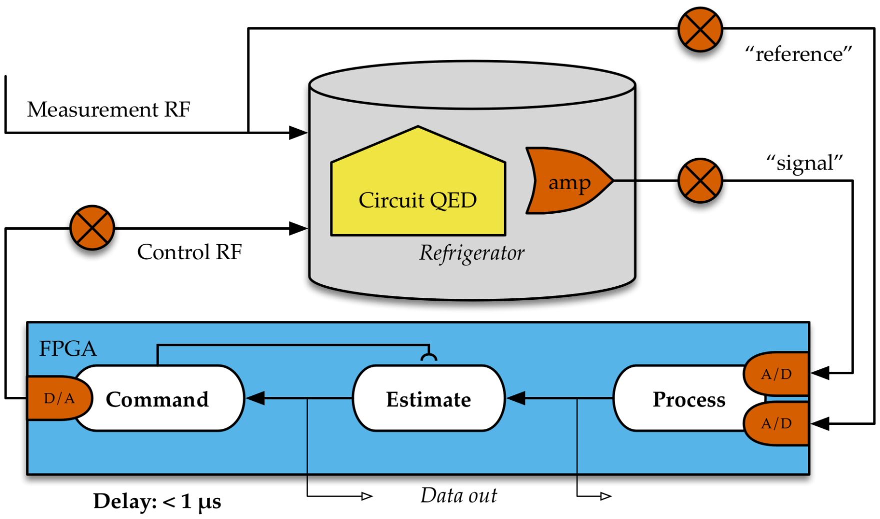


Figure 2-3: The diagram above highlights the connections between the quantum system and the FPGA when testing in the lab. The reference and signal RF tones are inputs to the FPGA in order to actually determine the appropriate control RF tone. This image is courtesy of Lisa Poyneer.

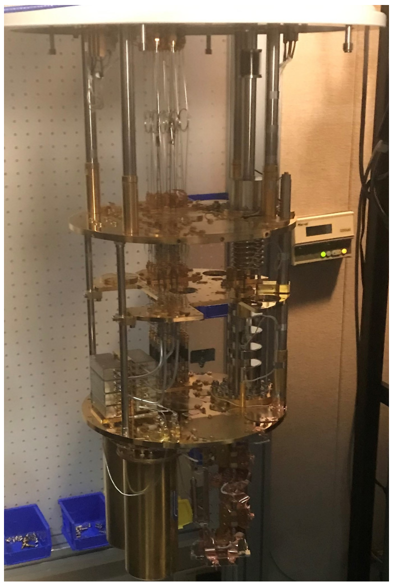


Figure 2-4: It is often useful to have an idea of the physical layout and form factor of the quantum system under testing. Each gold canister is a set of transmons inside the Bluefors dilution fridge.

addition to the control RF tone being modulated. The first set of laboratory experiments consist of comparing three data collection platforms. The Alazar digitizer, the

Integrated Innovations (II) FPGA, and the USRP. The Alazar provides the highest fidelity measurements but is not configurable for custom data processing in real time. It serves as the gold standard of comparison for the II and the USRP. The qubit system addressed for most of this work is known as Nash and sits inside one of the canisters within Jarvis. Figure 2-4 shows Jarvis during maintenance after it has been warmed up. Each gold canister contains one or more qubit systems.

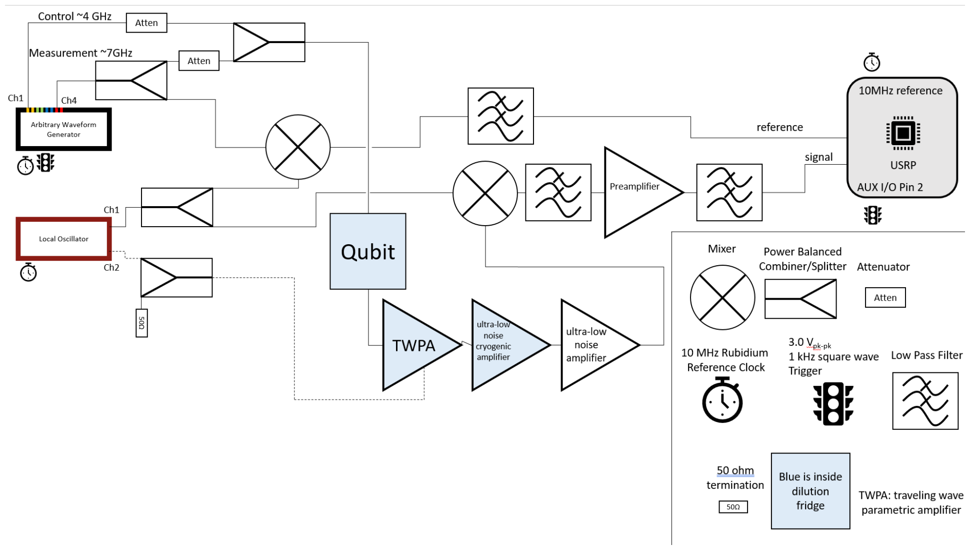

Control ~4 GHz
Atten
Measurement ~7GHz
Atten
Ch1
Ch4
Arbitrary Waveform Generator
Ch1
Local Oscillator
Ch2
50Ω
Qubit
TWPA
ultra-low noise cryogenic amplifier
ultra-low noise amplifier
Preamplifier
reference
signal
10MHz reference
USRP
AUX I/O Pin 2
Mixer
Power Balanced Combiner/Splitter
Attenuator
Atten
10 MHz Rubidium Reference Clock
3.0 V pk-pk 1 kHz square wave Trigger
Low Pass Filter
50 ohm termination
50Ω
Blue is inside dilution fridge
TWPA: traveling wave parametric amplifier

Figure 2-5: A more detailed layout of the experimental setup is helpful for drawing the correct conclusions, troubleshooting, and inspiring new experimental ideas such as changing the IF used and seeing the impact on performance. This image is courtesy of Spencer Tomarken.

Figure 4-16 has a more detailed breakdown of the experimental layout in the lab. The AWG is controlled via the lab computer using a program called Labber and generates the control and measurement RF tones. A low phase noise local oscillator drives the mixers so that the RF tones can address Nash. The signals coming out of Jarvis are amplified (both at cryogenic temperatures and room temperature) before being filtered and mixed down and being fed into the USRP for measurement and data processing. All the components share a 10MHz temperature compensated Rubidium clock and pulses are generated alongside a 1KHz square wave trigger signal.

2.2 Development Workflow

Creating applications and implementing algorithms for FPGA based systems follow a different development cycle than CPU applications. CPU programming at the most fundamental level is organizing processor instructions to fetch, process, and store data. FPGA programming involves implementing algorithms such that they can be synthesized in hardware from fundamental logic blocks. However, some traditional programming paradigms do carry over into FPGA applications. Modularity is one of those aspects that is consistent. A large part of the workflow for this thesis work involved modular creation and testing of portions of the FPGA application and algorithm before integration occurred. Another similarity can be seen in the user interface since that portion was implemented on a traditional CPU for ease of use. The methods used for this work revolve around a development flow which spans multiple tool chains. The tool chains used include LabVIEW (LabVIEW FPGA), Matlab (Simulink and HDL Coder), Python, Xilinx (Vivado), and Labber. This list while not exhaustive is relatively comprehensive.

2.2.1 System Modeling

With nothing more than the idea that the quantum state of the superconducting qubits can be estimated and controlled using an FPGA, algorithm creation and implementation cannot begin. The crucial first step is actually developing a deeper understanding of the quantum system itself. This process began by working with physicists to understand the signal characteristics going into the fridge as well as coming out in addition to any parameters set impacting the fridge unrelated to the signals. Following a high level understanding of the experiment, developing a simulation to help understand how different types of measurements perturb the system was paramount, thus simulation began in Matlab and Python. Using QuTiP, a high fidelity python based quantum simulation API, an intensive detailed quantum simulation of the system was created to model and verify those parameters and quantum behavior in response to different algorithms for sensing and control. A lower fidelity and less computationally intensive simulation was also created in MATLAB to facilitate faster prototyping but major milestones in progress that involved changing any fundamental parameter were invariably checked against the more precise simulation in QuTiP.

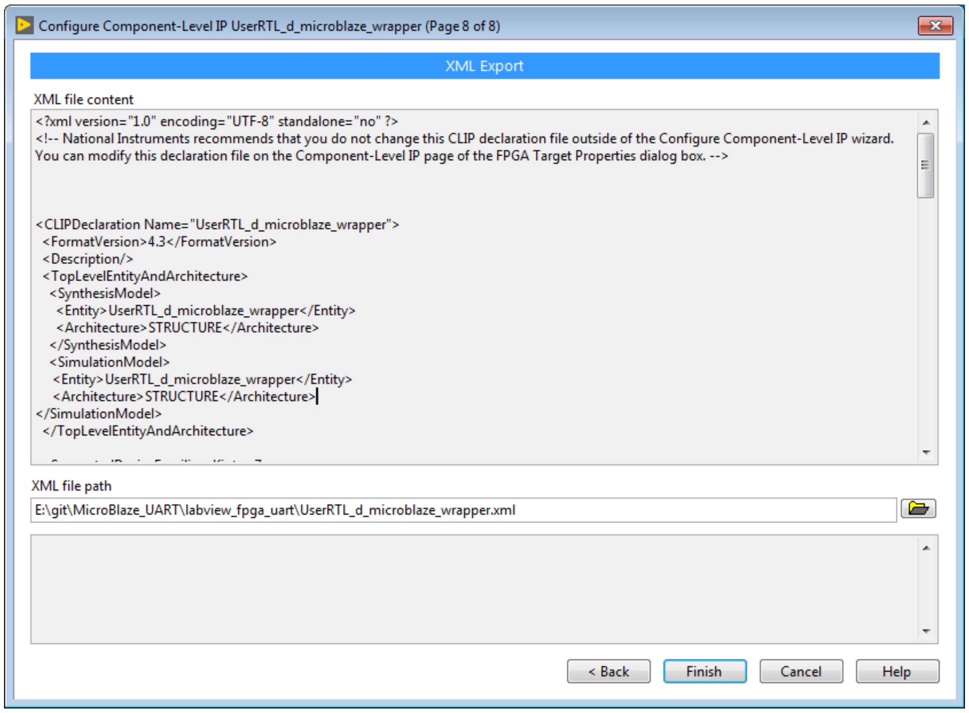
2.2.2 Algorithm Creation

The actual algorithm development (see Chapter 3) began by outlining the three stages of the computation: Process, Estimate, and Control. These three stages represent the high level goal of the algorithm to process the incoming RF signals to discern a phase difference (process), use that information to estimate the probability trajectory of the system (estimate), and the compute the appropriate control signal (control). The code implementation began in MATLAB with scripts operating on fabricated data. Once these MATLAB scripts were behaving as expected (in conjunction with QuTiP and MATLAB physics simulations), development moved to Simulink. Initially still using floating point values the Simulink implementation was compared to the MATLAB scripts implementation since the native data type used in MATLAB scripts is also floating point. Only once these agreed did the team move to fixed point Simulink implementation in order to more accurately represent what would eventually be put on the FPGA. This implementation was then compared to the floating point output and adjusted until a tolerable error was reached between the floating point and fixed point implementations. Throughout the process Simulink offered a wide range of tools for aiding in the algorithm creation including data dictionaries (custom data types), reference models (modularity), and HDL optimized functions (extremely useful in meeting timing).

2.2.3 Hardware Realization

Once a suitable fixed point Simulink model is achieved, the process of hardware implementation can begin. Using the HDLCoder MATLAB package, the team was able to generate VHDL that implements the fixed point functionality of the Simulink model.

During the generation process the user has the option to customize how the generated VHDL is formatted with everything from reset polarity options to adapative pipelining to automatic testbench generation. After the VHDL is generated, HDLCoder also provides statistics and analysis on the timing and estimated resource utilization for a given target device. However, the final synthesis, place and route, and device programing is completed using the native Xilinx compilation tools. For the II this happens in ISE while for the USRP this happens under the hood since it is insulated from the Xilinx compilation tools. Instead, the USRP uses LabVIEW FPGA as the development environment. The generated VHDL has to be modified slightly before being imported into the LabVIEW FPGA and compiled for the USRP.


Configure Component-Level IP UserRTL_d_microblaze_wrapper (Page 8 of 8)
XML Export
XML file content
XML file path
< Back
Finish
Cancel
Help

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!-- National Instruments recommends that you do not change this CLIP declaration file outside of the Configure Component-Level IP wizard.
You can modify this declaration file on the Component-Level IP page of the FPGA Target Properties dialog box. -->



<CLIPDeclaration Name="UserRTL_d_microblaze_wrapper">
 <FormatVersion>4.3</FormatVersion>
 <Description/>
 <TopLevelEntityAndArchitecture>
  <SynthesisModel>
   <Entity>UserRTL_d_microblaze_wrapper</Entity>
   <Architecture>STRUCTURE</Architecture>
  </SynthesisModel>
  <SimulationModel>
   <Entity>UserRTL_d_microblaze_wrapper</Entity>
   <Architecture>STRUCTURE</Architecture>
</SimulationModel>
 </TopLevelEntityAndArchitecture>
```

E:\git\MicroBlaze_UART\labview_fpga_uart\UserRTL_d_microblaze_wrapper.xml

Figure 2-6: This final stage of the XML Wrapper generation process in LabVIEW FPGA shows a preview of the generated XML indicated that the user could write the XML wrapper from the ground up, but to avoid time costing errors, this work simply used the native generation tools. This image is courtesy of John at fpganow.com

These modifications are to account for a lack of supported data types in LabVIEW FPGA as well as library structure issues since reference models manifest as external libraries when HDLCoder generates VHDL. However, even with these changes, the VHDL isn't ready to be integrated in the National Instrument's LabVIEW environment just yet. Figure 2-6 shows an example of the XML 'wrapper' that must be created for the VHDL to define the ports such that LabVIEW FPGA can interpret them correctly. This entire process involved quite a bit of trial and error, NI specifications research, and interfacing with NI support teams.

2.2.4 Analysis

The "final" step of the development workflow involved analyzing data collected either in a bench top fashion or laboratory fashion in order to validate (or invalidate) the current implementation. The analysis workflow began with the generation of the appropriate stimuli for the USRP, either internal signals, AWG signals, or quantum system output. The USRP would then log the input data as well as any processed data to disk. Developing this functionality is what is referred to as the 'LabVIEW Host' throughout this work. It offered real time data viewing, data saving utility, and tunable updating capabilities. In the future, it would also interface with the USRP outputs. The interaction with the physical experimental setup occurred by using Labber to drive the laboratory set up. Finally, MATLAB scripts were used to generate the plots in post experiment data analysis to determine the effectiveness of the implementation. The post processing comes in two flavors. Firstly, is simply plotting and formatting outputs saved by the USRP to ensure the right signal quality (SNR, spectrum shape, etc). Next is actually taking the inputs saved by the USRP and "playing" the values through the Simulink model. The fixed point Simulink will output values that should match the outputs saved by the USRP running in real time. This comparison allows for more fine tuning of parameters and bit widths. By using real data, bit starved operations and over-precise operations in the Simulink can be rectified quickly. Another aspect of the analysis is the comparison of the outputs generated by the II and Alazar versus the USRP. The goal of this work is to understand the capabilities the USRP has in quantum control experiments and validate it as a viable platform, and to do that it should be compared against industry standard tools using the metrics from above.

Chapter 3

Quantum Sensing and Control Algorithm (QSCA)

Controlling the the state of this work's quantum system is the end goal of the QSCA. The QSCA is the functionality that goes inside the blue square (see Figure 1-5 or 2-3). Another goal of this work was to improve and modify the QSCA to leverage any USRP specific properties while still maintaining portability across FPGA platforms. The QSCA has 3 core components: Process, Estimate, and Command. The Process stage detects a phase difference in the two input signals. Next, the Estimate stage applies LLNL proprietary algorithms to estimate the evolution of the quantum state probability distribution. Finally, the Command stage uses that estimate to generate the appropriate command signal in closed loop feedback. The first two stages (Process and Estimate) were part of the prior work done by the LLNL team before this work began. However, this work did contribute to the Command stage development as well as making any adjustments to the Process and Estimate stages in order to ensure functionality on the USRP.

3.1 Process

The Process stage of the QSCA begins the algorithm by implementing a moving periodogram, to measure the phase difference between the two signals, that expects two input waveforms from an ADC along with two tunable parameters. One parameter selects the window size of the moving periodogram and the other provides an overall phase offset (since the output of this block is a mangitude and phase). The general scheme leverages the fact that a predetermined IF of 25MHz is being used, thus the internal LOs are set to mix the input signal with two LOs at 25MHz but 90 degrees out of phase. The output is then averaged using a moving average whose window is determined by the tunable parameter. The resultant signals are sent into an optimized CORDIC algorithm block in order to compute phase and magnitude information. The key piece of information is the phase, but the magnitude information helps signal and synchronize the Estimate and Command stages (See 3.2 and 3.3) on when to start computing a motion update and command update, respectively. The phase calculated is initially in degrees, but some processing is required to make the values suitable for the Estimate stage, and as a result the final output was a normalized value from -1 to 1. This also required logic to handle phase wrapping issues after the CORDIC output but before heading into the Estimate stage.

3.2 Estimate

The Estimate stage, comprised of a state estimation algorithm, used a variety of tunable parameters and the instantaneous phase difference generated by the Process stage to calculate a quantum state probability distribution estimate. This estimate mapped to the trajectory of the qubit's state. The state estimation algorithm used was a bayesian tracker. Similar to a Kalman filter, the tracker had an internal model of the physics driving the quantum system based on system parameters such as measurement strength and qubit frequency. Each moving periodogram output was used to calculate the weight assigned to the likelihood of ground or excited. This information was then used in conjunction with the internal physics model to understand how the state changes due to natural evolution as well as the measurement back action. Finally, the internal state of the qubit is updated using a discrete state-space model of the system differential equation involving the Hamiltonian. This process occurs once every window as specified in the initial tunable parameter provided to the Process stage.

3.3 Command

Unlike the previous two stages, the Command stage was largely unimplemented before this work began. Thus, more time was spent exploring possibilities of how to generate the appropriate command signal in a closed loop situation. Two options stood out: a simple pseudo-PLL or state feedback. The latter posed a more significant challenge with unproven worth but interesting possibilities. However, given the time constraints, the former was implemented instead. A basic PLL, or phase locked loop, operates as control loop designed to keep two signals (a reference and a measured) in phase with one another. It does this with three basic components: a phase comparator, a low pass filter, and a voltage controlled oscillator (VCO). In this work the phase comparator is implemented as a mixer with a fixed LO, the low pass filter is still a low pass filter, but the VCO is replaced with the dynamics of the quantum system since the oscillation frequency of the qubit's resonance is dependent on a DC voltage.

The general scheme involved using the estimate as a trajectory to which a correction needed to be applied (due to the phase difference at the beginning of the QSCA). This was achieved by multiplying the input to the Command block with the output of a local oscillator at the desired frequency which served as a reference. This product was then filtered to remove the high frequency component. Finally a gain was applied and a nominal offset added to generate the final command output. An diagram detailing the structure can be found in Figure 3-1.

3.3.1 Mixer

The first stage of the Command block is a mixer. The mixer expects two inputs: a local oscillator (LO) and the Estimate stage output. Both signals are at the same Rabi frequency $f_r = 625$ kHz. The output of the mixer will produce a signal that

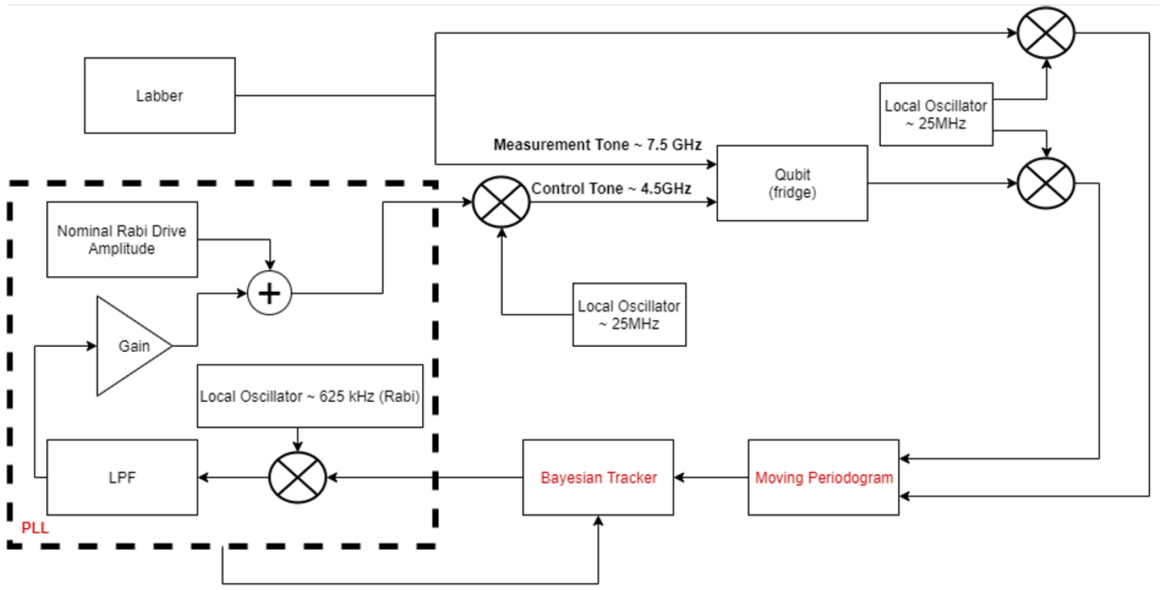


Figure 3-1: This block diagram details the QSCA section structure with an emphasis on the command section structure. The three key stages of the command section are the mixer, low pass filter, and the gain/offset.

has two frequencies $f_1 - f_2$ and $f_1 + f_2$. Figure 3-2 demonstrates the operation of a mixer like the one implemented in Simulink. In this implementation the LO has been configured such that $f_1 = f_2 = f_r$ providing an output signal at $2f_r$ with a DC offset that is related to the phase difference of the signals. A 0° phase difference will produce a positive DC offset, 180° of phase difference will produce a negative DC offset, and 90° will produce zero DC offset. Thus, since the Estimate block produces a signal that needs to be phase-locked to the LO, the Command block needs produce non-zero command signals whenever there is a non-zero phase difference. This is achieved by intentionally offsetting the LO output by 90° before mixing with the Estimate output. Thus, the mixer stage generates a signal at $2f_r$ with a DC offset proportional to phase difference of the estimated trajectory of the quantum system and the reference phase provided by the LO.

3.3.2 Filter

Following the mixer, the Command block needs to extract the DC offset from the signal. This comes in the form of a low pass filter (LPF). The design and imple-

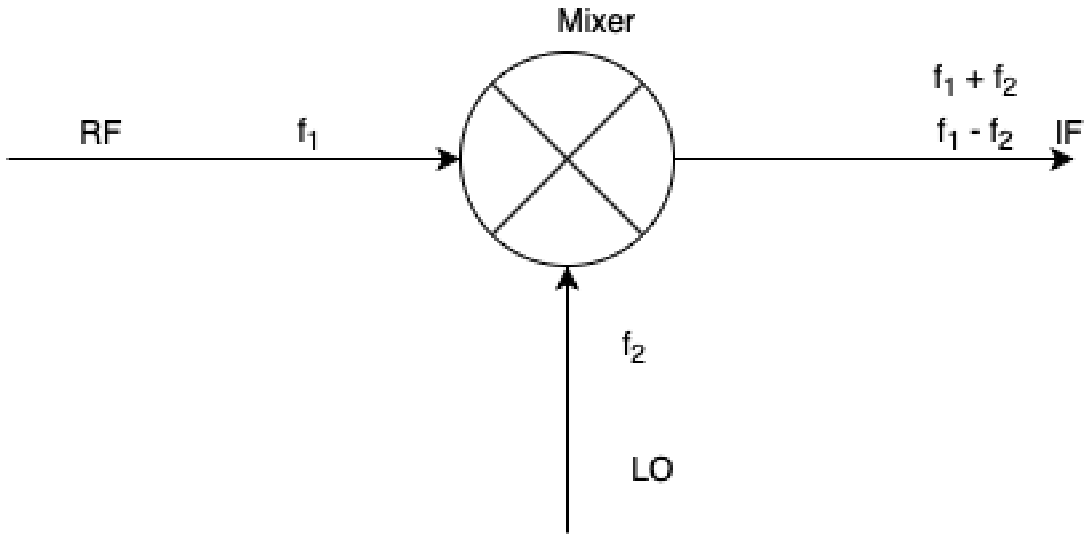


Figure 3-2: The image above shows the basic operation of a mixer. For this portion, the RF line is the estimate from the Estimate stage and the LO is the a local oscillator at the Rabi frequency of 625kHz.

mentation of the LPF provided a wide array of design tradeoffs. The first tradeoff involved selecting if the LPF should be digital or analog. The entire system up until this point (aside from the radio peripherals of the USRP) was implemented digitally, so the kneejerk response was to continue in a digital fashion. However, low latency was a huge issue, and digital filters have notoriously longer latencies for a given level of attenuation. Thus, analog filters seemed to be the way to go, but an even larger issue of flexibility of design put a stop to that. This coupled with the fact that with careful tuning digital filters can perform as good as their analog counterparts settled the first design tradeoff and work continued toward implementing a digital LPF. The first step was to define a theoretical bound of performance as a relationship between the LPF bandwidth and latency. The bandwidth in this case is the cutoff frequency of the LPF while the latency can be measured through the 10% to 90% rise time of the step response of the LPF. Using a single stage, first order LPF to model the theoretical best case scenario, the solution to the differential equation describing the system is

$$V(t) = V_0(1 - e^{-\frac{t}{\tau}}) \tag{3.1}$$

where the time constant of the LPF defined as

$$\tau = \frac{1}{2\pi f_c} \tag{3.2}$$

which is a necessary component in the expression of the rise time which can be reached by solving equation 3.1 for time.

$$\begin{aligned} t &= -\tau ln(1 - \frac{V(t)}{V_0}) \\ t_{10\%} &= -\tau ln(1 - .1) = \tau(ln(10) - ln(9)) \\ t_{90\%} &= -\tau ln(1 - .9) = \tau ln(10) \\ t_{rise} &= t_{90\%} - t_{10\%} = \tau ln(9) \end{aligned}$$

Substituting equation 3.2 into the rise time expression leaves an equation for the rise time as a function of the cutoff frequency f_c

$$t_{rise} \approx \frac{.35}{f_c} \tag{3.3}$$

Recall section 3.2 and the update latency of the Estimate block as dependent upon the tunable parameters selected in the Process stage. The latencies can be $20, 40, 160, 320$ ns and in order to actually compile and synthesize at 250MHz or 200MHz (the clock speeds of the II and USRP respectively), the latency allowed for the Estimate block must be either 160 ns or 320 ns. This corresponds to a minimum cutoff frequency of $f_c = \frac{.35}{320ns} = 1.09375$ MHz for the LPF. Since the Rabi frequency is set at 625 kHz the signal after the mixing stage will be at 1.25 MHz giving a little more than 150kHz of breathing room for the attenuation.

With a minimum cutoff frequency on hand, the design turned toward another set of tradeoffs: filter type. Even within the realm of digital filters, there is a wide variety of filter types that can be mostly categorized as IIR or FIR where for the most part IIR filters are extremely analogous to their counterpart analog implementation while FIR filters are more "natively" digital. This is because IIR filters require feedback to implement while FIR filters are simply a weighted sum of previous data values.

Filter Order / Cutoff Frequency	1 Hz	100 kHz	750 kHz
16	3.693	3.7016	3.7017
256	0.5449	3.7353	3.7073
1024	.0692	3.7057	3.7453

Table 3.1: This table demonstrates the relationship between filter order and cutoff frequency by measuring the ratio of the stopband ripple to the DC offset, a metric that is desired to be as small as possible.

The magic happens when picking the weights (this is true for both IIR and FIR) [8]. IIR filters provide improved performance in the form of stop band attenuation and latency at the price of wider ranges of instability and non-linear phase relationship with the output. Within the realm of IIR filters there were 4 types that were considered: Chebyshev Type I, Chebyshev Type II, Elliptical, and Butterworth. The two Chebyshev filter implementations are mirrors of each other in terms of the presence of stop band ripple versus pass band ripple while the elliptical draws a compromise between the two with sharp roll off but ripple in both the stop and pass bands. A final decision hasn't been made on which filter implementation will be used as the work continues, though the stability and linear phase relationship of FIR filters is attractive and the initial performance metrics seem to be sufficient for this use case.

Another important factor explored within the filter design was filter order. In its simplest form a higher order filter is similar to refiltering a signal that has already been filtered. There are optimizations done in order to reduce the overhead of actually just refiltering a signal, such that a higher order filter performance better than two cascaded filters, but the notion is similar. As a result the delay of a higher order filters is naturally higher with benefit of smaller stop band ripple as illustrated by Table 3.1. Figure 3-3 compares the performance of two FIR filters as a function of filter order.

Luckily, the entire filter design process is heavily reliant on computational tools within the MATLAB environment since it provides a variety of filter design tool kits in addition to optimized filter blocks intended to generate VHDL using HDLCoder. The filterDesigner GUI (see Figure 3-4) in MATLAB allows the user to select the type

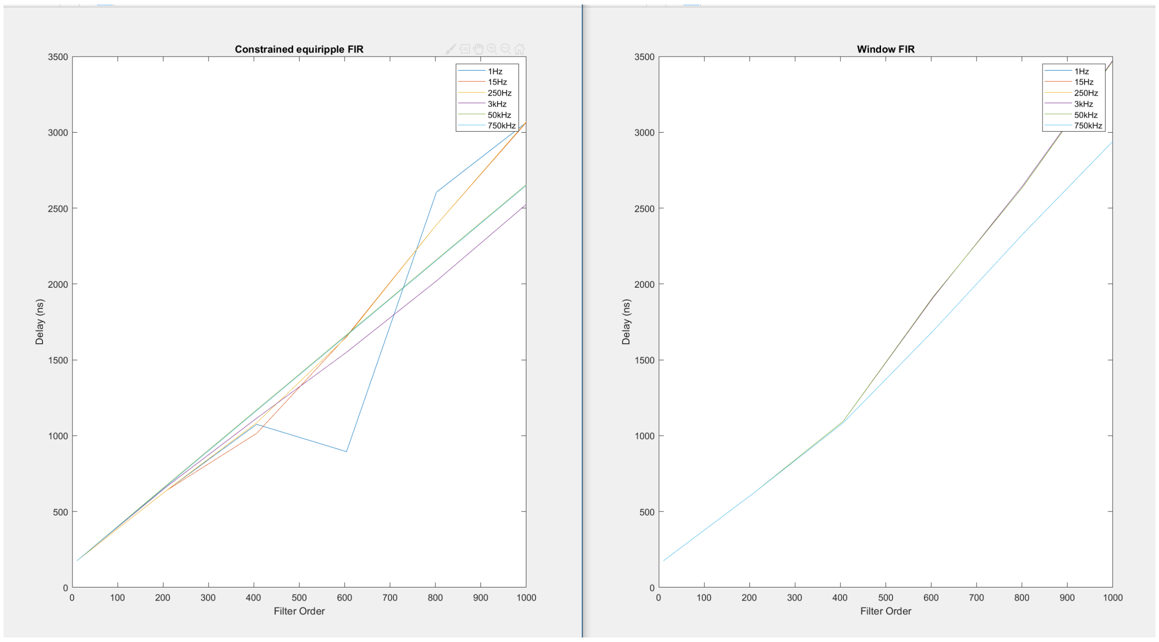
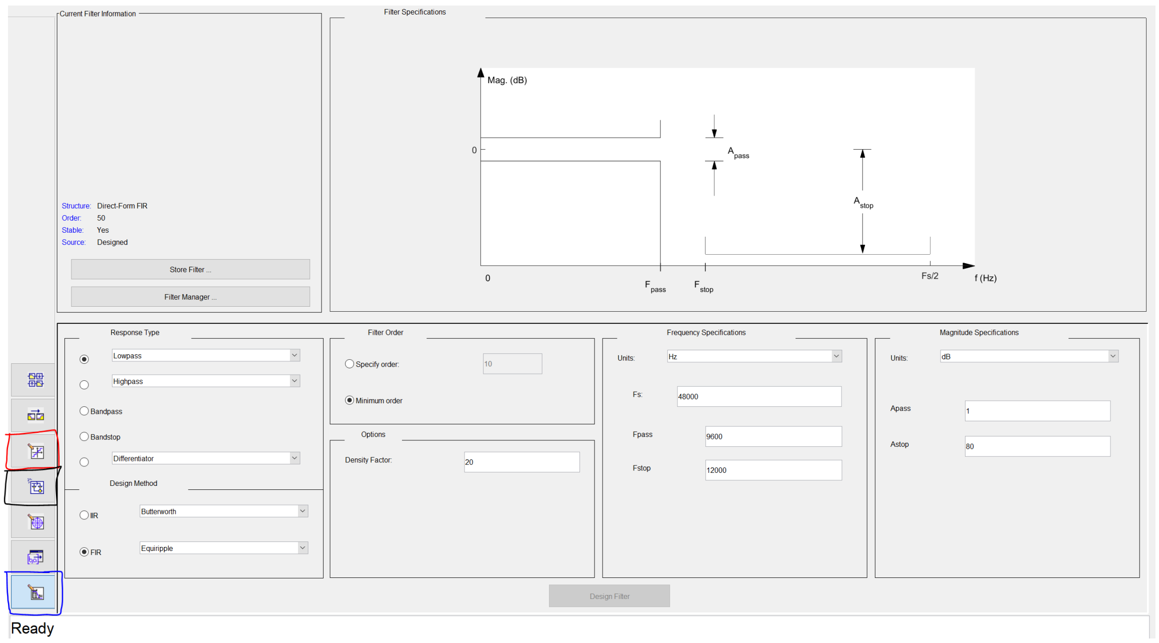


Figure 3-3: These graphs show the relationship between the delay of two different types of filters as a function of filter order. On the left is the constrained equiripple approach and on the left is the window FIR filter approach. Both graphs have 20 data points with filter order ranging from 10 to 1024.

of filter (FIR or IIR), the precise implementation (everything from an equiripple to a high order Chebyshev), as well as the relevant parameters for that implementation. This goes hand in hand with the HDLCoder side of things since the filterDesigner can produce a coefficient list. On the surface, HDLCoder's optimized filter blocks only allow the user to change a few exposed parameters, but the main benefit of these optimized blocks is the implementation within is user editable. Thus, using the coefficients generated from the filterDesigner and the HDL optimized layout of registers, multiplies, and accumulates a decent filter can be realized in hardware via MATLAB and HDLCoder.

The final trade off worth noting is one that came up in other parts of the algorithm development as well: division. A lot of the filter implementations utilized a division stage which could implemented in one of three ways: natively, as a look up table, or a bit shift. Each method has its benefits and drawbacks, though in order to meet timing constraints the go-to method has been a look up table. For the time being a bit shift was used and the divisors were forced to powers of two. Figure 3-6 and 3-5



Structure: Direct-Form FIR
Order: 50
Stable: Yes
Source: Designed

Response Type
[✓ Lowpass]
Highpass
Bandpass
Bandstop
Differentiator

Design Method
IIR Butterworth
[✓ FIR] Equiripple

Filter Order
Specify order: 10
[✓ Minimum order]

Options
Density Factor: 20

Frequency Specifications
Units: Hz
Fs: 48000
Fpass 9600
Fstop 12000

Magnitude Specifications
Units: dB
Apass 1
Astop 80

Figure 3-4: The image above shows what the filter design tool UI allows a user to control. Everything from bit widths on coefficients to pole placement tools are available.

show an IIR and FIR, respectively. In both cases the "Desired" used for comparison is a baseline floating point moving average filter. Both track relatively well, but surprisingly the IIR seems track better which is surprising given the baseline moving average filter can be expressed as a basic FIR filter.

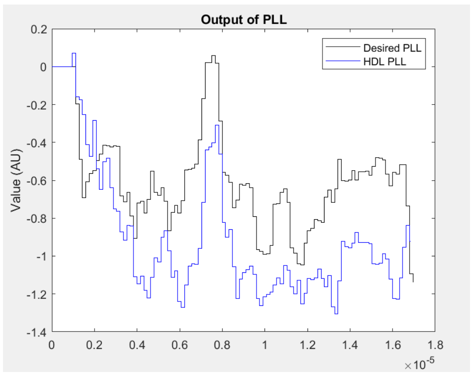


Figure 3-5: Equiripple FIR

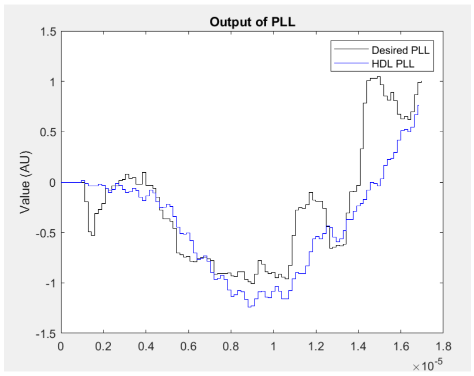


Figure 3-6: Butterworth IIR

3.3.3 Gain and Offset

After the LPF stage of the Command block, the final touch to generate the command signal is the gain and nominal offset. Both of which are tunable parameters designed to amplify and offset the command signal to the appropriate value before the USRP radio peripherals transmit the signal back to the quantum system. Simulations suggest a gain value of 0.5 provides an appropriate value for the command signal, however, tuning gain values also depend on the filter selected. For example, Figure 3-6 and 3-5 also have a difference in gain, yet show similar tracking to the desired PLL value.

Chapter 4

USRP Validation

Keeping in mind that the overarching goal of this work is to determine the viability of the USRP as a FPGA platform for quantum sensing and control, the process of validation boils down to ascertaining the capabilities of the USRP, comparing them against the required specifications, and searching for alternative implementations to deal with how the specifications are structured or leveraging other benefits of the USRP to overcome any shortfalls. This chapter examines the capabilities of the USRP and features a case study of a testbed for the output of the initial process step of the QSCA in addition to an analysis of the RF front end capabilities when collecting actual data from the quantum system.

4.1 Baseline Capabilities

Before any "real" work could begin with the USRP and the QSCA, an understanding of the USRP's general development environment and capabilities was required. This meant understading exactly how the USRP functioned and any quirks in the behavior that could pose problems in the future.

As with most programming languages, beginning with a "Hello World" of sorts is always a good place to start. For the USRP, that meant controlling the front panel LEDs. Figure 4-1 provides a snippet of the code used to control the LEDs on the USRP. This proved useful later as an indicator for when a new bit file had

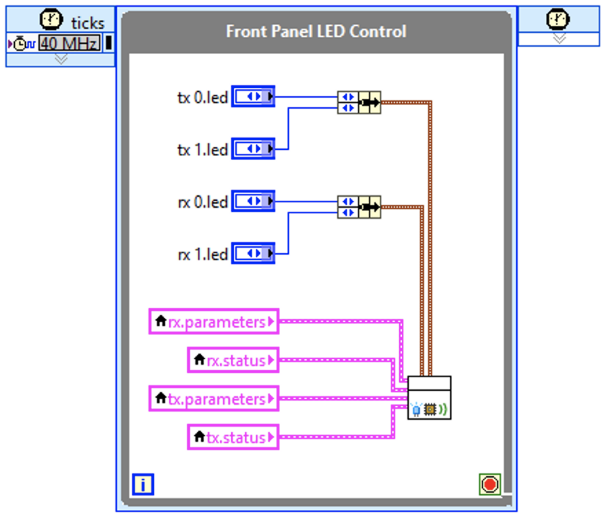


Figure 4-1: Controlling the LEDs on the USRP with signals from both the host and with internal triggers proved to be a useful starting point for understanding the LabVIEW Block Diagram flow.

been programmed, the receive data path was active, or if the USRP was responding at all. Utilizing the USRP for increasingly complex mathematical operations was the next logical step. Beginning with simple addition of hard coded numbers, this work required building up to more complex operations such as reading values from a file and performing simple DSP (FFT) operations in real time. The final checkpoint was ensuring that the capability surrounding data writing was well understood in order to take finalized data values and send them to the host before saving to disk. This required careful consideration on the part of parallel processing since such large amounts of data were being fed back to the host. Figure 4-2 shows one of the later implementation of the saving utility that saved in a burst fashion.

All of these capabilities avoided the more complicated DSP blocks such as digital down conversion as well as interacting with the RF front end in an effort to simplify and accelerate the learning curve of the development environment. In sections 4.3 and 4.4 the DSP blocks and analog RF hardware are utilized and discussed.

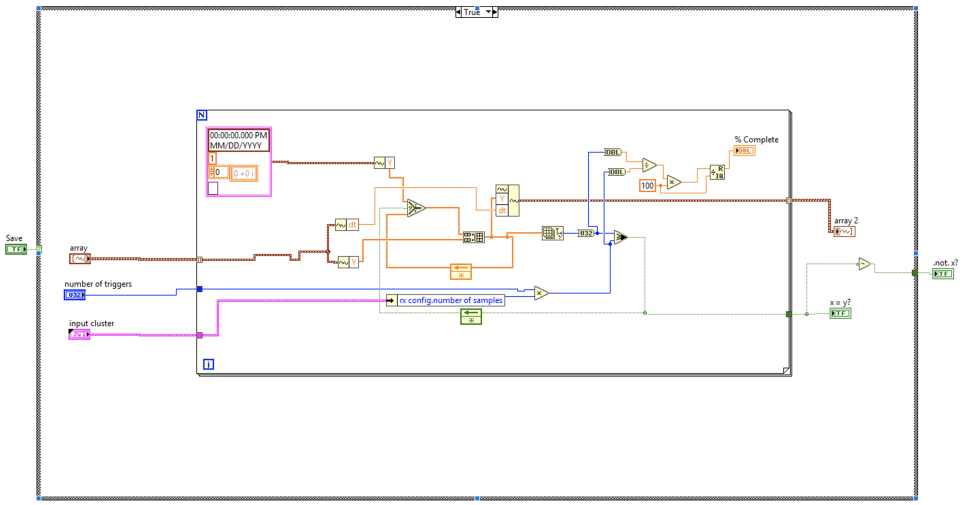


Figure 4-2: This LabVIEW block diagram snippet to save data proved to be a crucial step in testing the QSCA

4.2 Case Study: Moving Periodogram Testbench

With a basic understanding of the USRP's capabilities and development environment, this work moved on to a case study of the first portion of the QSCA on the USRP.

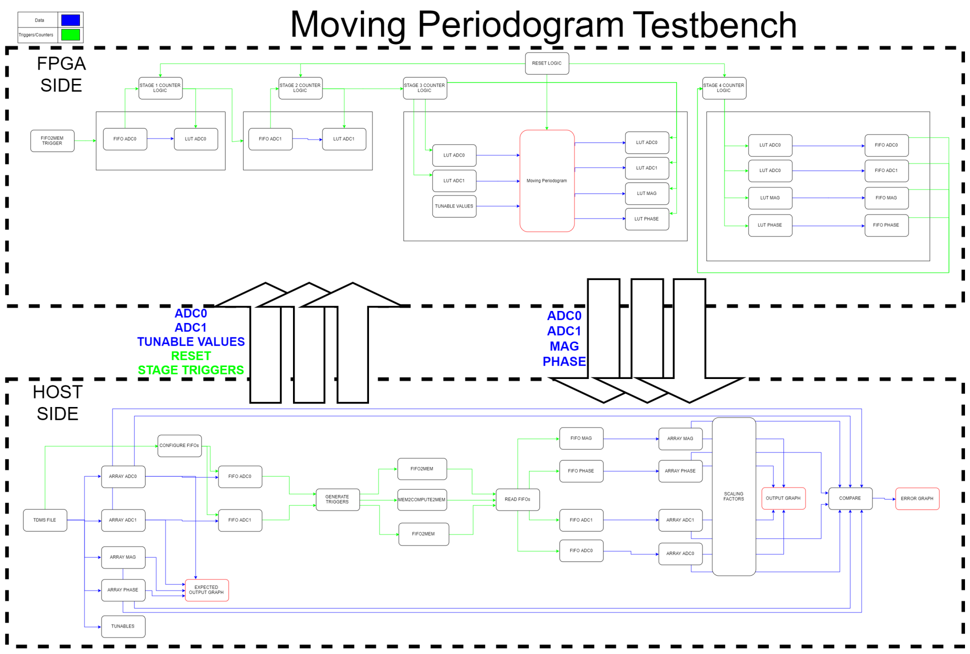


Figure 4-3: This diagram outlines the signal paths and scaffolding used to test the moving periodogram on data from a file. It traces out the flow of data and triggers between host PC and USRP target

The moving periodogram served as the first important calculation step of the algorithm and ensuring the USRP could reliably produce the correct output is the subquestion that this case study answered. Even without any DSP and RF front end interaction, validating the moving periodogram as a case study proved to be extremely valuable in learning even more of the intricacies of the USRP development environment. Moreover, the actual process of migrating the existing Process implementation to the USRP and validating its functionality offered the additional benefit of providing insight to the Simulink/HDLCoder workflow and how it interacts with the USRP.

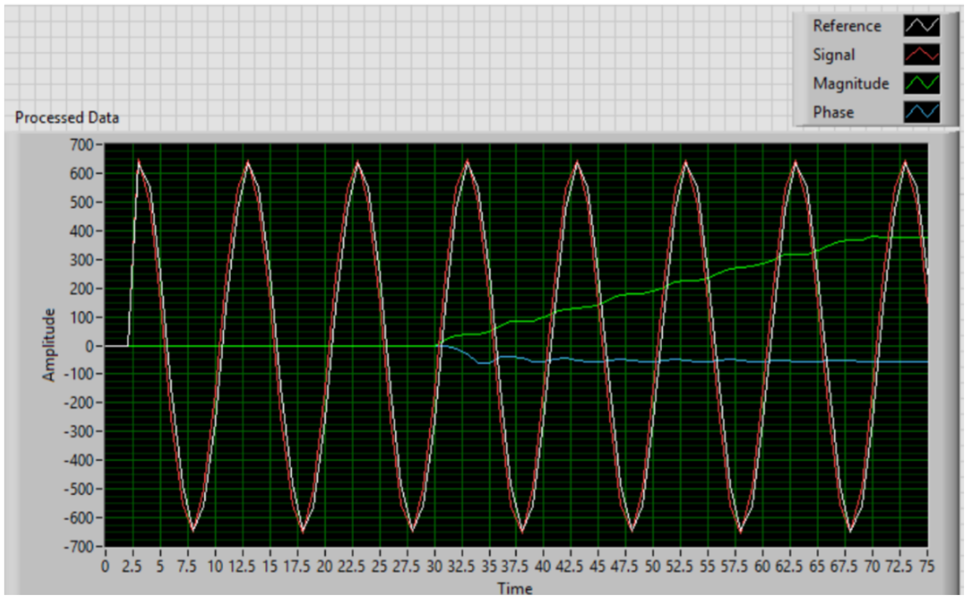
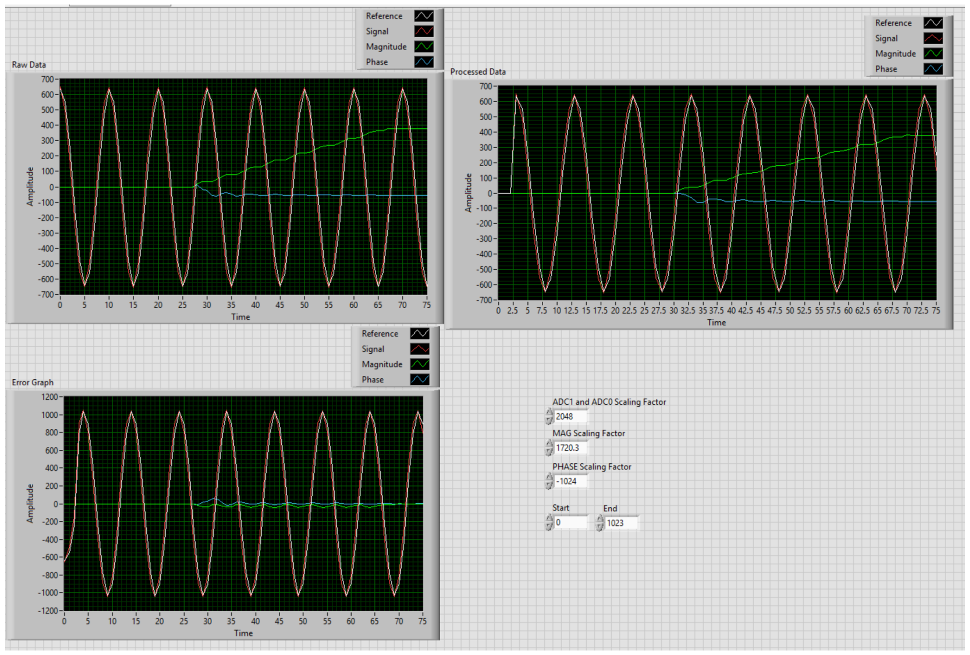


Figure 4-4: Many plots like this one were generated using data collected by the II as input to the USRP to determine if the calculations would match as expected. The plot shows the two inputs (reference and signal) along side the two outputs (magnitude and phase).

The test bench was run on both real data (open loop, collected in the lab using the II) and AWG data collected by the II. Using these signals as a stimulus to the Process step, the USRP would ideally produce an output that matched what the II produced. The block diagram in Figure 4-3 outlines the scaffolding required to implement a test bench for the Process block. The green arrows indicate control signals such as reset signals and state machine transition signals. The blue arrows represent data transfer either as arrays of data that can be post processed offline or singular important tunable values that affect how the algorithm functions. The red boxes are key blocks for user input or output such as plots in a real time viewer.



ADC1 and ADC0 Scaling Factor 2048
MAG Scaling Factor 1720.3
PHASE Scaling Factor -1024
Start 0
End 1023

Figure 4-5: The scaling factors were brought out as a tunable parameter to determine what adjustments needed to be made when switching from the II to the USRP.

The results of the case study are summarized by Figure 4-4. Using data collected by the II under AWG stimulation as input for the USRP, the USRP was able to match the expected (II) calculated output. However, when data was "played" through the USRP's implementation of the moving periodogram a few general issues presented themselves. First and foremost was the issue of quickly moving data from the host to the USRP and vice versa. The USRP development environment provided a wide range of tools to transfer data from a memory mapped register system to a DMA FIFO transfer engine. Each method had its own pros and cons. However, in an effort to focus on the validating if the USRP was calculating the correct values, a workaround was employed. The data was split into small chunks, approximately 1023 fixed point values, and the data was transferred to the USRP one point at a time before being run through the Process portion of the QSCA. The output was then stored in memory on the USRP before being transferred to the host PC one point at a time. From there the host could construct the waveform and compare the value to what the values the II had calculated. These tradeoffs and issues stemmed from the USRP's architecture, however a few issues were actually due to mishandling of the moving periodograms's specifications.

While the values matched at the end as in Figure 4-4, this was only after dealing with a myriad of issues. One of the confounding issues was data initially looking completely flat. All the curves (reference, signal, magnitude, phase) seemed to be at zero. This issue was traced to a difference in scaling factors. The II's ADCs were configured with a specific value for the scaling and this was maintained when the II data was used as input for the USRP. In order to deal with this, the scaling factor for each type of signal was included as a parameter in the test bench as can be seen in 4-5. Another issue that came up was a horizontal (time) offset as in 4-6. This first manifested as out of phase sine waves on the signal and reference lines. Upon further investigation the issue was traced to two separate issues. The less consequential one was the inclusion of 28 cycles of "dead" time at the start of the file that needed to be accounted for in order to match the II's calculations. The second issue was a trigger misalignment meaning the data was not moving through the algorithmic pipeline when intended. This trigger alignment would prove to be an issue throughout this work because of the tight timing constraints. Luckily all these issues were caught relatively early in the process and thus, the USRP was able to pass the first major test and allowed the work to move on to bench top testing using signals from an AWG.

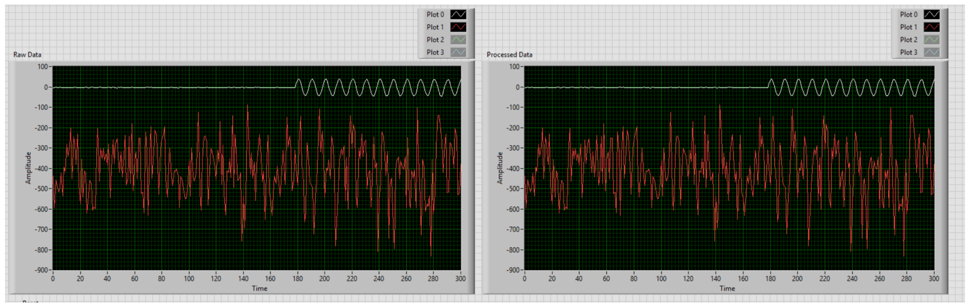
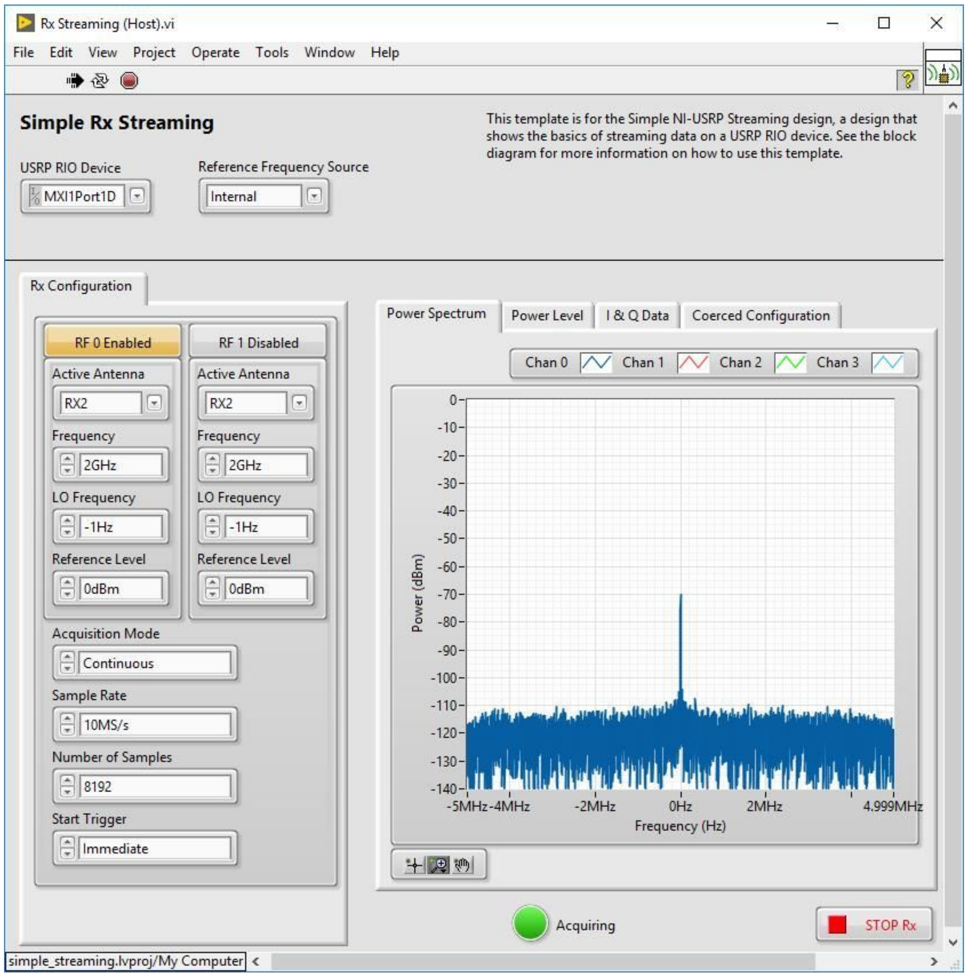


Figure 4-6: The trigger alignment issue first presented itself in the case study and continued to be a challenge throughout this work.

4.3 Benchtop Testing

Once the moving periodogram output was validated using static data read from previously saved AWG and quantum data, the next step was to begin to understand how the actual RF components functioned. The process actually began with heavy interactions with National Instruments in which they provided two things: a detailed circuit schematic of the RF front end and sample LabVIEW program to setup the receive and transmit channels and view incoming signals in real time (see Figures 4-7 and 4-8).



Simple Rx Streaming

This template is for the Simple NI-USRP Streaming design, a design that shows the basics of streaming data on a USRP RIO device. See the block diagram for more information on how to use this template.

USRP RIO Device: MXI1Port1D
Reference Frequency Source: Internal

RF 0 Enabled — Active Antenna: RX2; Frequency: 2GHz; LO Frequency: -1Hz; Reference Level: 0dBm
RF 1 Disabled — Active Antenna: RX2; Frequency: 2GHz; LO Frequency: -1Hz; Reference Level: 0dBm

Acquisition Mode: Continuous
Sample Rate: 10MS/s
Number of Samples: 8192
Start Trigger: Immediate

Figure 4-7: The basic stock example of data viewing that LabVIEW provided for streaming data often dropped samples which was unacceptable given the phase measurement done by the moving periodogram was sensitive to such errors. Issues like this would not be as noticeable in a scenario focused on wireless communications research (the original intension of the USRP)

Both of these pieces of information brought different issues to light when working with the USRP. The circuit schematic in Figures 4-8 revealed a few troublesome filters that could potentially eat into the SNR of the data coming out of the quantum system. While multiple band pass filters didn't pose an issue, there was a concern that the 80MHz low pass filter had a cutoff that was a bit too close to some of the data which had components as high as 60MHz in testing with the II. Also, the sample LabVIEW application seemed to have issues dropping data as it was transferred from the USRP to the host. This was the same issues seen in the previous section and ignored, but that was no longer possible since the inputs would now be coming from real RF signals.

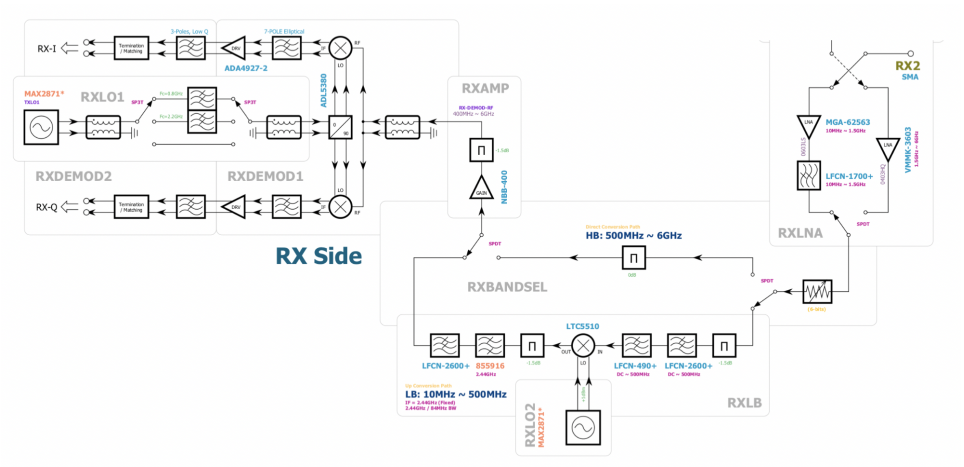


Figure 4-8: The analog front end schematic brought the multitude of internal filters to light and caused worries about interfering with or degrading the SNR of data collected using the USRP

4.3.1 Baseline Data Capture

The first task of the bench top testing process was to ensure the USRP could reliably capture data from an AWG and save it to disk. This meant modifying the sample LabVIEW application from a real time viewer to a data collection application. The save functionality would play a key role in ensuring a uniform data format that could be read in by MATLAB for further offline processing and output validation. After iterating through a few different LabVIEW paradigms including producer/consumer, a state machine, and serial processing this was achieved. The producer/consumer paradigm is demonstrated in Figure 4-9

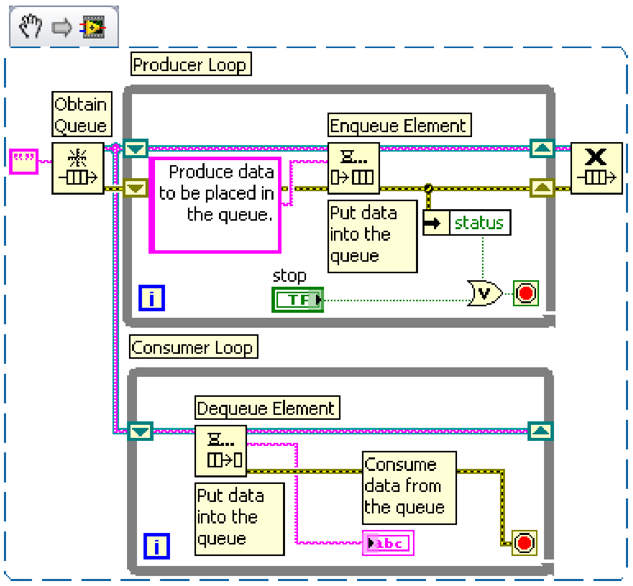


Figure 4-9: This code snippet shows the popular parallel data processing paradigm used in LabVIEW known as producer/consumer. By using FIFOs to communicate between parallel loops, data can be processed by two different rate loops.

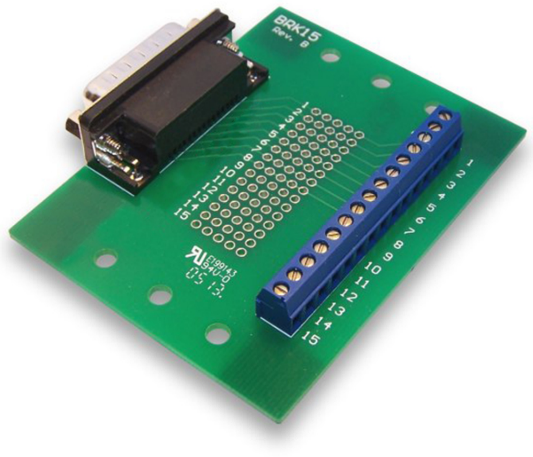


Figure 4-10: The atypical connection pattern to accurately trigger on on an external signal began with a GPIO Expansion Kit. Courtesy of Ettus Research.

4.3.2 Moving Periodogram

With data reliably being sent to the USRP and USRP output being saved by the host, the work moved on to actually validating the USRP's ability to take in data from an AWG, send it to the Process step of the QSCA, and save the output to a file for processing and validation. The first major obstacle in this step was correctly connecting all the necessary ports and cables. The USRP required an external 10 MHz reference clock, the signal and reference input lines, the trigger input. The trigger input proved to be the most difficult. Requiring a breakout board (Figure 4-10) and a series of adapters to connect to a typical SMA cable to provide a 1 kHz trigger (to mimic what was used in the actual lab setting), which dictated the burst data read frequency.

After successfully connecting the USRP to all the required ports, the next challenge came in actually ensuring the USRP was looking at the right IF for data. Unfortunately, overloaded terminology made this quite difficult with no clear way of setting the desired center frequency of the receiver. Carrier frequency, center frequency, local oscillator (LO) frequency, and just frequency were all parameters in the "basic" LabVIEW application for receiving and transmitting signals. This resulted in quite a few data sets that had PSDs similar to Figure 4-11. After a few conversations with National Instruments engineers to get exact definitions for the parameters, this work moved one step closer to the next goal of detecting a clear IF as in Figure 4-12.

The next challenge was ensuring no data values were being dropped because if any values were dropped, the output would no longer be aligned with the trigger and none of the analysis would be valid or useful. In order to ensure this didn't happen the initial basic LabVIEW application was modified to adopt a producer/consumer framework and non-essential operations removed from the base application including removing the digital down conversion (DDC) block which caused issues later on during lab testing. The DDC does the job of ensuring an incoming signal is precisely mixed down to IF (in the case of this work, 25 MHz). The hardware mixes the signal down to a level close to IF, and the DDC uses built-in calibration data to finish the job.

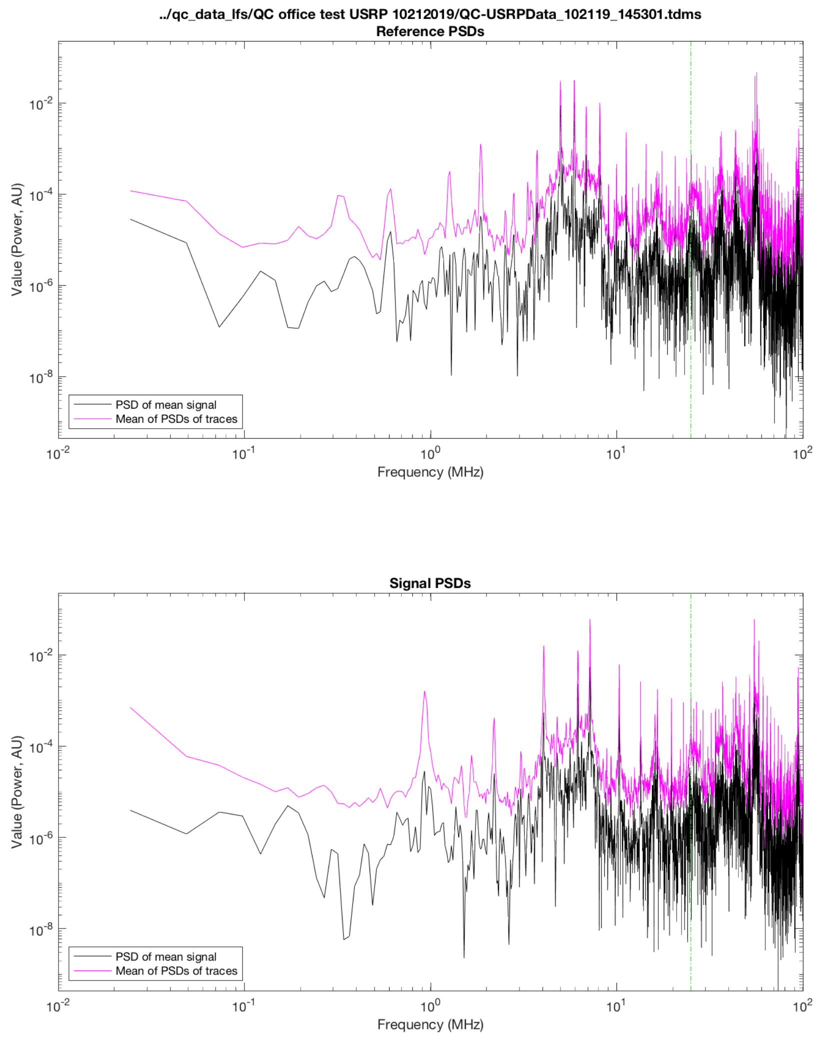


Figure 4-11: While experimenting with the right center frequency/carrier frequency quite a few bench top data collections resulting in PSDs like above with no clear IF peak.

Using the calibration values is also known as IQ impairment correction. While, the DDC is a costly block in terms of latency, it turns out to be necessary in order to get quality data from measurements. After some trial and error on what portions of the application were essential to the proper calculation and configuration of the USRP, data collected by the USRP was finally aligned with the trigger and tuned to the IF of 25 MHz resulting in a PSD as in Figure 4-12.

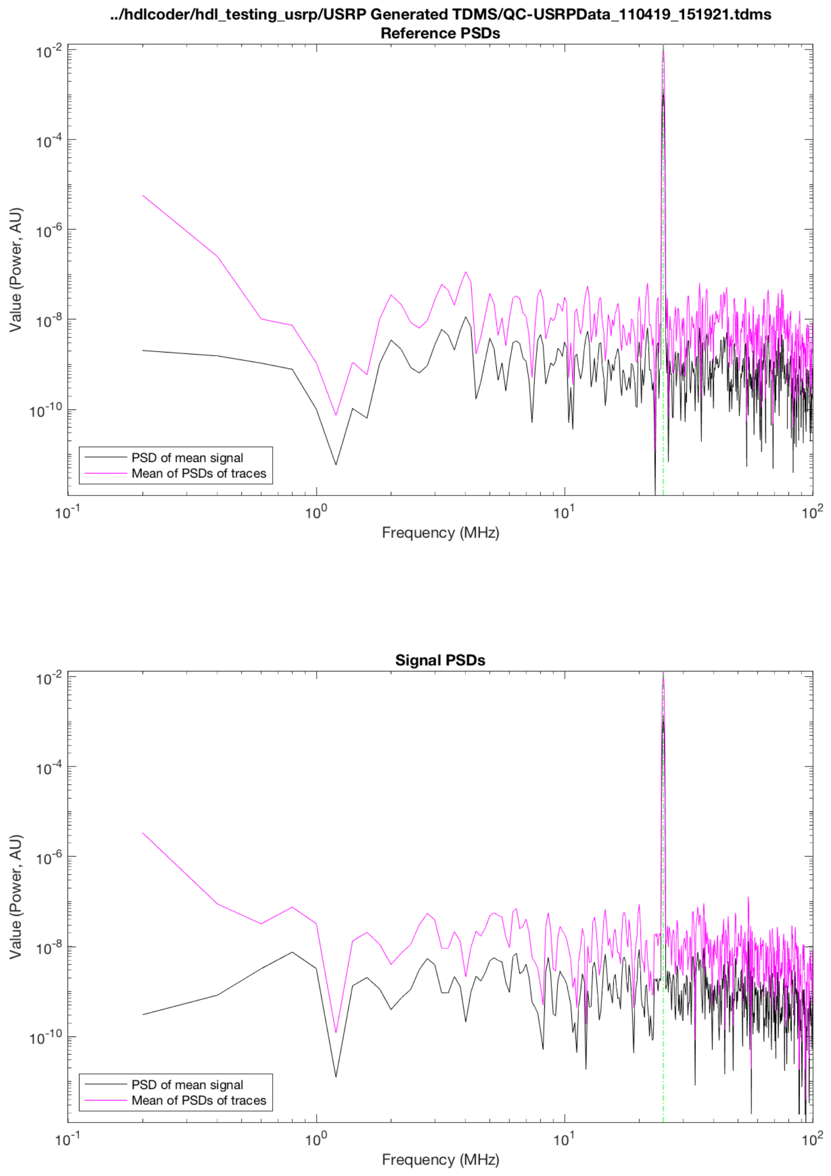


Figure 4-12: First major milestone of the USRP collecting data with a clear IF peak at 25 MHz.

4.3.3 Bayesian Tracker

After the USRP's implementation of the moving periodogram was reliably outputting data that matched the of the II FPGA, work moved on to the Bayesian Tracker. This portion of the work had stalled on the II because of a mismatch between the output of the II's Estimate stage and the predicted output by MATLAB, which until this point had been an exact match. Diagnosing this issue for the II meant re-synthesizing at

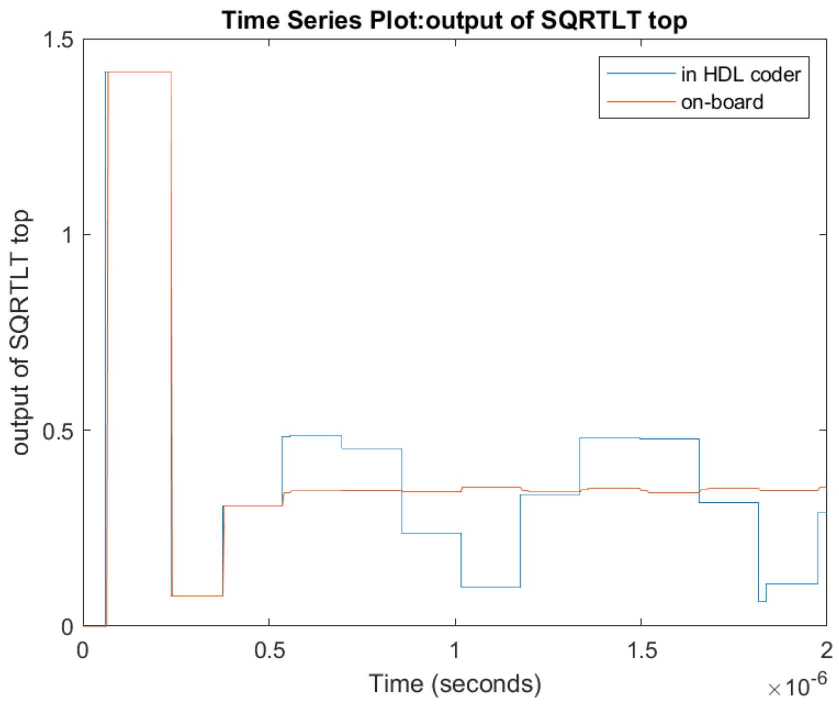


Figure 4-13: Bayesian Tracker updates for first three values and then stops on II courtesy of Lisa Poyneer

multiple different stages of the Bayesian Tracker algorithm and re-testing. This was a cumbersome process that narrowed the error down to a the output of a single square root look up table. However, this didn't provide any clue as to why the mismatch was arising here.

Moving to the USRP and running the same workflow did provide some insight however. When the same moving periodogram + bayesian tracker implementation as was used on the II, was put onto the USRP, the calculated output was the same. This consistent but incorrect calculation can bee seen in Figure 4-15 and indicates an issue upstream of the hardware.

Since the workflow for the II and the USRP after the VHDL generation step is vastly different, and yet the the issue nearly identical, this added credence to the notion that the issue was with the way HDLCoder was converting the Simulink implementation of the algorithm to VHDL. Another possible issue would be with the

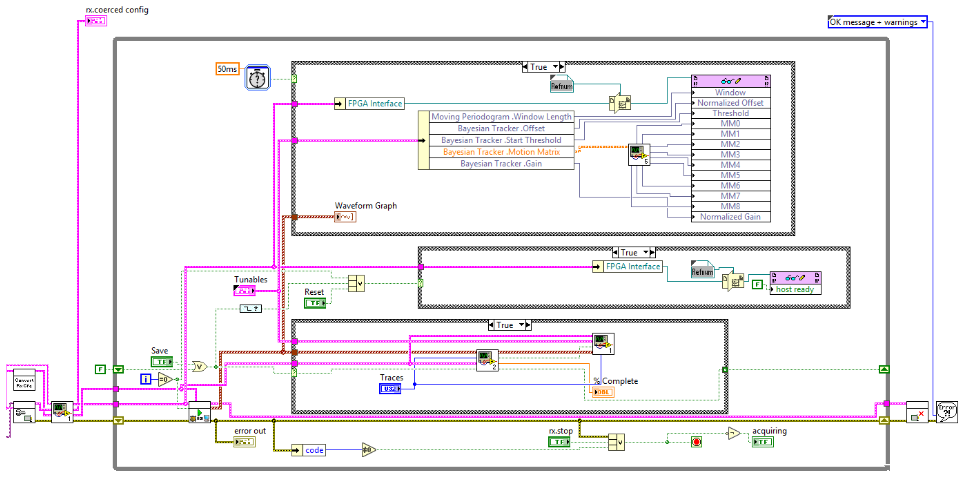
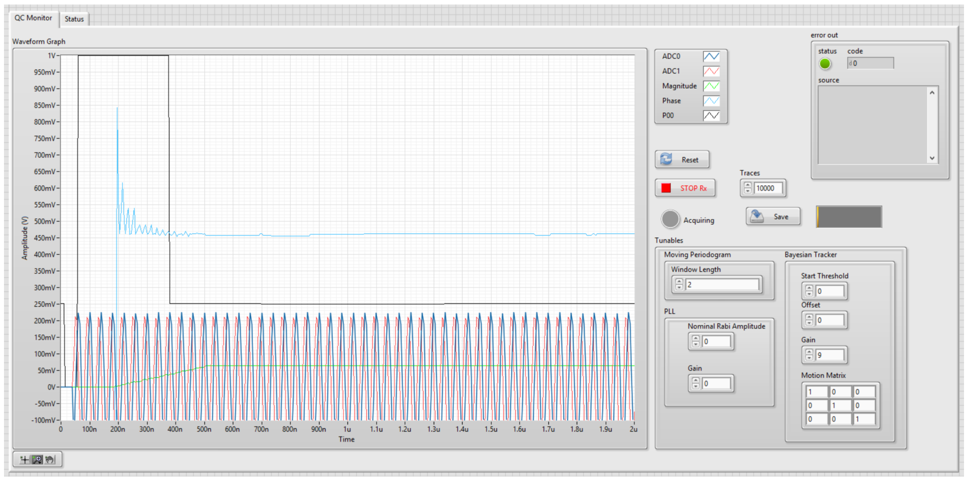


Figure 4-14: LabVIEW code that implements LabVIEW Host for moving periodogram and bayesian tracker.

algorithm itself, but the validation of the Simuilink version with the floating point MATLAB script version seems to discourage this explanation.



code	0
Traces	10000
Window Length	2
Nominal Rabi Amplitude	0
Gain	0
Start Threshold	0
Offset	0
Gain	9

Motion Matrix

1	0	0
0	1	0
0	0	1

Figure 4-15: Bayesian Tracker updates for first three values and then stops on USRP

4.4 Quantum System Testing

Once testing began in the lab, the data gathered would provide the clearest possible picture of how well the USRP would perform in relation to the II and the Alazar digitizer. While the initial plan was to work with the signal directly and mix down using USRP, the plan was changed to keep the external heterodyne for two key reasons: in order to do a more direct comparison of the the performance of the II and the USRP as well as to avoid any issues with the upper limit of the USRP's input frequency range of 6 GHz.

Setting up the USRP with the quantum system for data collection was a similar process to that of the bench top testing. The 10 MHz reference clock, signal and reference lines, and trigger all had to be connected as before. The trigger again posed issues since the USRP expected a TTL signal on the signal input and the trigger used up to this point had an amplitude of 1 Volt. Figure 4-16 provides a more detailed layout of the connections

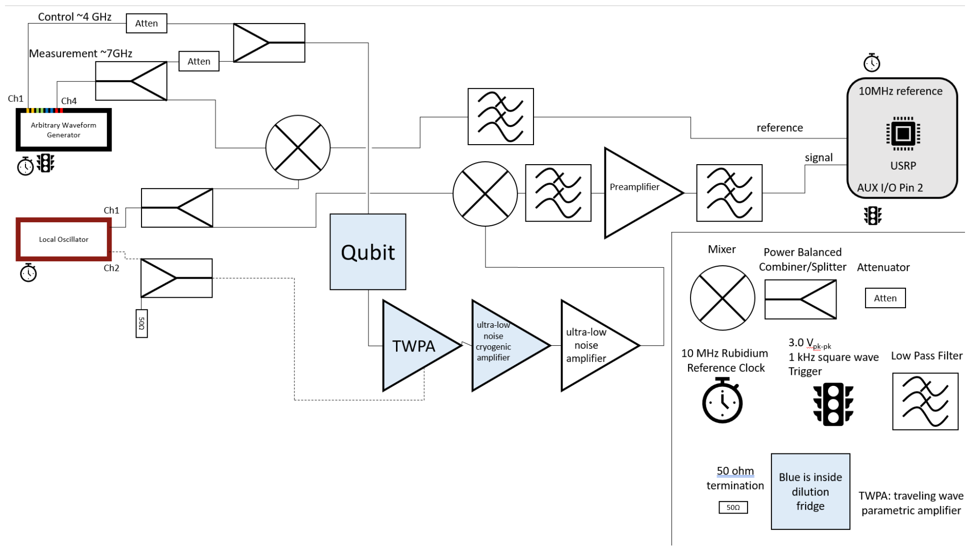
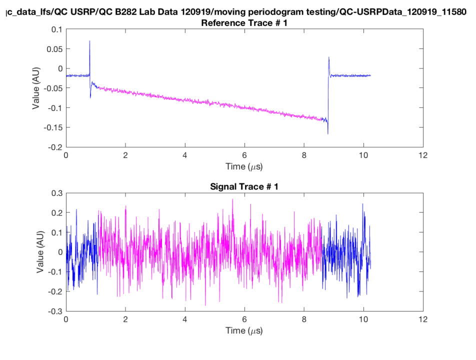

Control ~4 GHz
Atten
Measurement ~7GHz
Atten
Ch1
Ch4
Arbitrary Waveform Generator
10MHz reference
reference
signal
USRP
AUX I/O Pin 2
Preamplifier
Ch1
Local Oscillator
Ch2
Qubit
50Ω
TWPA
ultra-low noise cryogenic amplifier
ultra-low noise amplifier
Mixer
Power Balanced Combiner/Splitter
Attenuator
Atten
10 MHz Rubidium Reference Clock
3.0 V$_{pk-pk}$ 1 kHz square wave Trigger
Low Pass Filter
50 ohm termination
50Ω
Blue is inside dilution fridge
TWPA: traveling wave parametric amplifier

Figure 4-16: A more detailed layout of the experimental setup is helpful for drawing the correct conclusions, troubleshooting, and inspiring new experimental ideas such as changing the IF used and seeing the impact on performance. This image is courtesy of Spencer Tomarken.

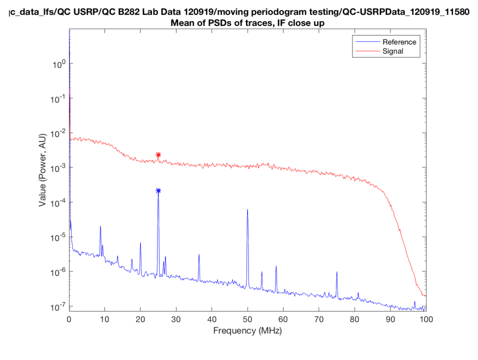
The quantum system is stimulated using a Keysight AWG controlled through Labber. This provides the control and measurement RF tones at various power levels configured to set the initial state of the qubit to ground or excited. While the end goal of this work would require a weak measurement to disturb the quantum system as little as possible, a range of power levels are checked to determine at what measurement strength the signals coming out of the quantum system are usable. This determination is made based on a variety of metrics including SNR and noise floor.

4.4.1 Baseline Data Capture and RF Characteristics

When moving into the lab, two key issues became a concern: maximizing the SNR as well as reducing the delay of the QSCA's USRP implementation. In order to focus on the latter issue, an attempt to reduce the amount of scaffolding code of the simple streaming example application begun. No direct measurement of the latency of the application before and after removing the scaffolding code was conducted, however a heuristic was developed instead. If data ever became misaligned with the input trigger, this indicated the latency of the implementation was too long. Latency also increased as more data per trigger was collected or an increased amount of triggers were collected. Thus, this work can claim that the more data that could be collected without trigger misalignment, the better the latency of the application. Before any modifications, were made to the base application, 512 data points per trigger could be collected just one time before the misalignment occurred. While by the end, 2048 data points with 10,000 triggers could be collected without a trigger misalignment. During this process, a majority of the code was safely removed without impacting performance, but there was a key portion that added both a significant delay and a key performance boost. This was tied directly to the former of the two issues: maximizing the SNR.



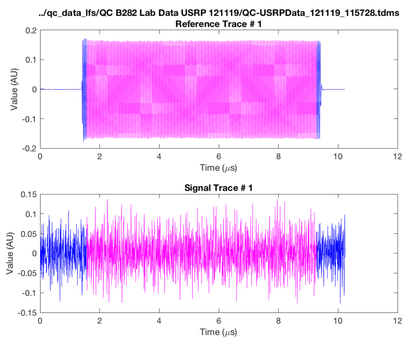
(a)



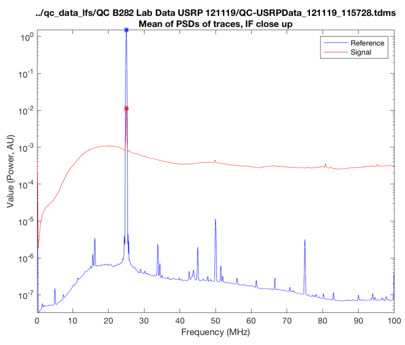
(b)

Figure 4-17: (a) This example trace shows what the USRP data would look like without the DDC. (b) When computing the PSD of average data collected by USRP when there is no DDC present, the IF is greatly attenuated

Maximizing the SNR was achieved via the direct digital down conversion. It allowed the USRP to maintain an SNR comparable to that of the II (see Figure 4-20b) while removing the DDC completely distorted any data captured. Figure 4-17a and 4-17b shows an example trace without the DDC running on the USRP. The IF is still there but greatly attenuated and now a low frequency component dominates the signal (see the AC coupling phenomenon of the noise floor).



(a)



(b)

Figure 4-18: (a) Sample trace of USRP collected data with DDC (b) Reintroducing the DDC and calculating the PSD of average data collected by USRP shows a clear IF.

Figure 4-18a and 4-18b shows how reintroducing the DDC vastly improves the performance of the USRP. The sample trace now more closely resembles a typical trace from the II or the Alazar and the 25 MHz IF is clearly visible above the noise floor. The main reason for this gap in performance is actually due to the fact that the DDC includes a step for IQ impairment correction that is tuned to a set of values unique to the layout of the USRP's analog front end. Each time a trigger is received, the IQ corrections are applied and the data is calibrated to the "true" value.

4.4.2 USRP Signal Quality

After successfully and consistently being able to capture data with the USRP in the lab, the next step in developing a clear picture of the USRP's RF characteristics is to take a few key metrics on the data gathered: SNR, noise floor, and temporal PSDs. The SNR provides the most direct quantitative way to compare the USRP's performance to that of the II, the Alazar, and any other platforms in the future.

Table 4-19a and 4-19b provide a snippet of the trend of the power SNR values for a range of measurement strengths. As expected, the SNR decreases with weaker measurement strengths as the system approaches the quantum noise floor. See the appendix for more fine grained table of SNR values. The SNR values provide a quantitative measure of the performance, while looking at traces also provides a qualitative idea on the performance based on the presence of any quantization in the output. This again matches the II in performance with the II achieving a 16.98 dB SNR at 260 mV and a 1.35 dB SNR at 40mV on the same system as the USRP. The Alazar also performance similarly with a 10.9 dB SNR at 7.6dBm input and 10.1 dB at 4.8 dBm input. However, since the Alazar cannot run the real time control loop, it is not a viable device and serves only to ensure that the data gathered by the II or USRP is sufficiently usable which the SNR measurements seem to support.

Measurement Power Level	SNR
260 mV	16.65 dB
120 mV	4.33 dB
40 mV	1.35 dB

(a)

Measurement Power Level	SNR
260 mV	17.17 dB
120 mV	4.57 dB
40 mV	1.37 dB

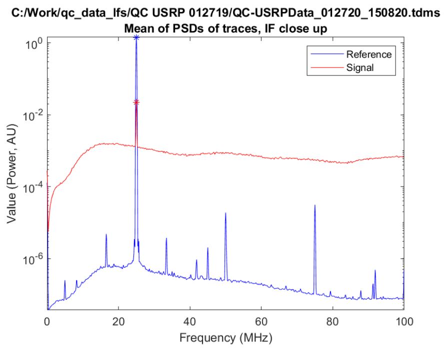
(b)

Figure 4-19: (a) Ground State SNRs (b) Excited State SNRs

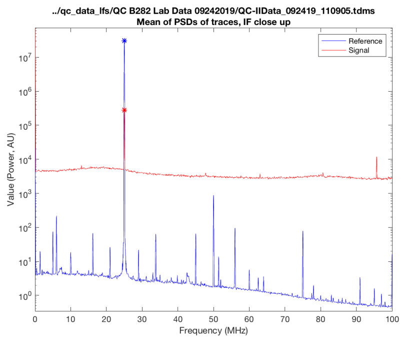
The next feature to look at is the noise floor of the data. In order to do this, this work focuses on the power spectral density (PSD) of data traces collected. By taking the average of all the PSDs a clear shape and peak appears around the IF of 25 MHz. The noise floor of the data collected had a characteristic "bump" at frequencies below IF as in Figure 4-20a. The II on the other hand as a relatively flat noise floor as in Figure 4-20b. This is most likely attributed to the AC coupled nature of the some of the components in the analog RF front end of the USRP. Overall, signal quality of the USRP seemed to be sufficient relative to the II to continue testing.

4.4.3 IQ Populations

After running the first round of tests to determine the signal quality and RF characteristics of the USRP, the next step is to determine if the calculations the USRP



(a)



(b)

Figure 4-20: (a) The PSD of the USRP shows a clear noise floor due to AC coupling (b) The II noise floor is relatively flat.

conducts produce data comparable to that of the II or Alazar. This is done by plotting IQ populations. Recall that the first stage of the QSCA, the Process, generates a phase and magnitude for each trace of data collected. By taking two sets of strong measurements, one at excited and one at ground, an IQ population histogram can be generated. The IQ populations are based on the entire pulse width (8 microseconds for most of the USRP data), but the USRP is producing a weak measurement trace in real time as in Figure 4-21.

The sharpness of the IQ populations generated depend heavily on the measurement strength. Looking at Figure 4-22a and 4-22b it is clear that at 40mV, the populations are bleeding into each other while at 260mV they are much more defined. This of course comes at the cost of disturbing the quantum system more and increasing the rate of decoherence.

Figure 4-24a and 4-24b demonstrate two sets of well defined IQ populations collected using relatively strong measurements. However, the populations are not in the same place on the complex plane, demonstrating a key issue: the populations are not guaranteed to be centered around the x axis, an orientation that would be useful when calculating the angle between the two populations. On the II, this also posed a problem, and in order to fix it a tunable rotation offset was created. The tunable parameter would allow the user to change the rotation offset after taking an initial measurement to determine how much of a rotation was required. This tunable would

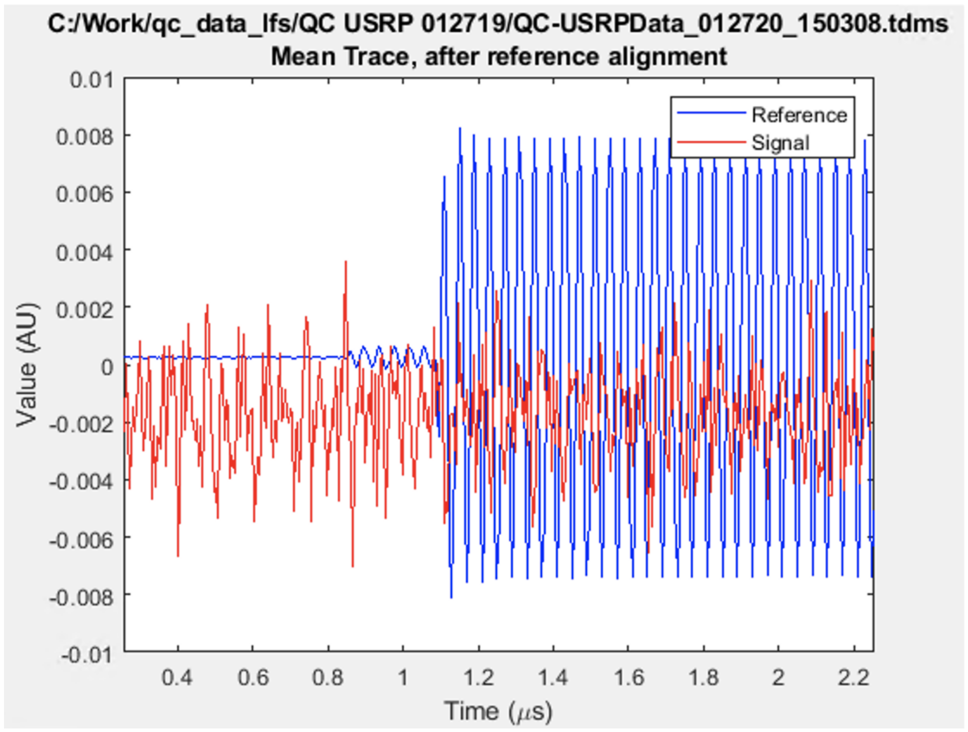


Figure 4-21: Average reference and signal trace taken by the USRP with a weak measurement at 340 mV

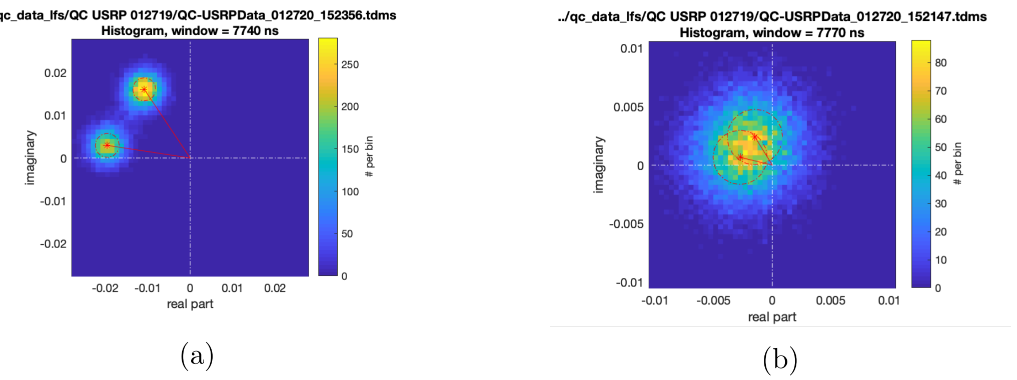
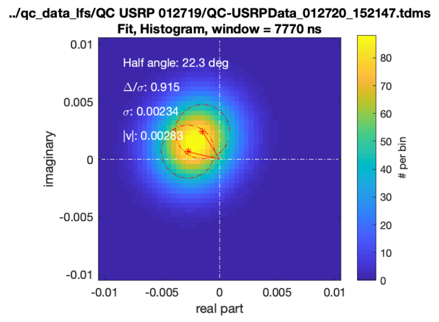
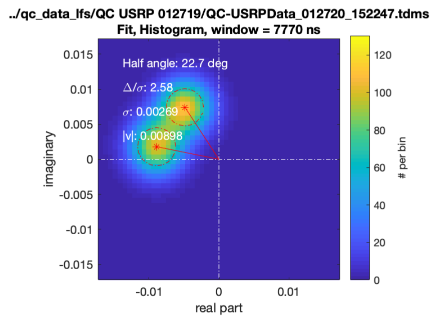


Figure 4-22: (a) 8 microsecond pulse of phase data collected by the USRP at 260mV measurement strength with 10000 traces collected and processed to extract IQ populations. (b) Same setup as (a) but with a measurement strength of 40mV.

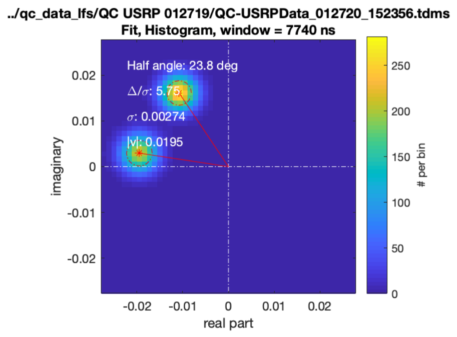
need to be recalibrated upon each new start up but not in between individual data collection runs. On the USRP, the issue was confounded by the fact that each reset of the local oscillator would change the rotation of the populations. Furthermore, each new reset of the application also reset the local oscillator inside the USRP, and since the USRP application was reset periodically after each data collection in order to ensure no trigger misalignment, each data set would have a different population rotation. The solution to this was to provide two signals of a known phase difference before each data collection to allow the USRP to calibrate itself, similar to the IQ calibration that occurs in the DDC. This means a real time tunable is necessary to counteract the rotation of the populations. This tunable alongside the window length tunable of the moving periodogram allows the data going into the bayesian tracker to be properly formatted and streamline calculations down the pipeline. Figure 4-23 shows 3 different data sets and the corresponding half angle measurements (post processed in Matlab). That half angle measurement is what the tunable would ideally account for in real time.



(a)

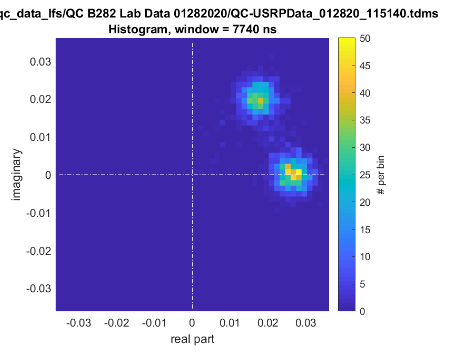


(b)

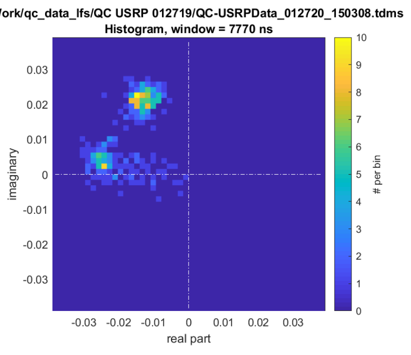


(c)

Figure 4-23: (a) Same as Figure 4-22b but with explicit angles called out. (b) Similar but at 120mV measurement strength. (c) 260mV measurement strength



(a)



(b)

Figure 4-24: (a) 8 microsecond pulse of phase data collected by the USRP at 340 mV measurement strength with 1000 traces collected processed to extract IQ populations. (b) Same setup as (a) but at 320 mV

Chapter 5

Alternate Technologies

LabVIEW and the USRP platform aside, there are other tools that should be analyzed alongside the USRP rather than analyzing the USRP's capabilities in isolation (which only paints part of the picture of which tools are best suited for quantum sensing and control). An analysis of comparable options is necessary to fully trust any conclusion about the viability of a platform like the USRP.

5.1 CPU vs ASIC vs FPGA

While the focus of this work has been validating the USRP as a FPGA platform, it is worth explaining why the scope is restricted to FPGAs in the first place. CPUs and ASICs offer benefits above FPGAs in a variety of cases. CPUs (and microcontrollers) are much easier to reprogram and test. They offer extremely fast iteration on development cycles. ASICs offer superior power and performance capabilities. The bulk of the work occurs in the design and optimization phase of the ASIC, and once the design of the ASIC has been finalized, no CPU will be able to match the performance of the chip produced. However, each of these devices have their drawbacks. An ASIC produces such amazing performance at the price of little to no reprogramming functionality. If a new parameter needs to be introduced, an ASIC has no way of incorporating that. The elevated cost, in both time and money, is another issue that plagues ASICs. For an application that is still development such as this

QSC algorithm an ASICs would have involved millions of dollars plus years of testing and tape out before a trusted final product could be produced. While a CPU offers great prototyping speed through easy programmability and lower cost to implement, a CPU based approach would suffer from performance issues when trying to meet the rigorous specifications of a quantum experiment like the one relevant to this work in terms of speed of calculation. Thus, an FPGA seems to be a logical choice. FPGAs offers more reprogramming functionality and lower than an ASIC cost but do require more effort and time to reconfigure than CPU or microcontroller. While they are not as robust as ASICs, FPGAs do provide enough speed and performance for this application, far beyond any CPU which also explains their middle ground price point, more expensive than a CPU but cheaper than ASIC.

5.2 FPGA Platforms

This section explores a few established methods for implementing tasks like quantum sensing and control that are FPGA-based. Two main methods will be discussed in the section: high level synthesis and HDL. Each have their own pros and cons that vary from project to project. Under the lens of this work, a hybrid implementation of the two made the most sense, leveraging the Simulink/HDLCoder workflow but also requiring some VHDL tweaking in order to ensure functionality across platforms. One less discussed approach is to start with just the bare bones FPGA chip and build up a custom analog front end, write custom data communication protocols (for data to move from the analog front end to FPGA as well as to get data off the FPGA back to the host), and build a native host application from scratch. This time and labor intensive approach seems to ignore possible existing technologies that could be leveraged in favor of avoiding any external learning curve. With a knowledge of RF circuits, Python (or similar language), and Verilog/VHDL, one could theoretical tackle the entire project. However, using existing solutions to portions of the problem greatly reduce development time. Reinventing the wheel, while impressive, is unnecessary in this case.

5.2.1 High Level Synthesis (HLS) Approach

Starting close to the top of the stack, is the HLS approach. The general theme of these methods is to abstract away as many of the unnecessary hardware details as possible. These approaches vary in just how "high level" they actually are, with some requiring little to no knowledge of the underlying hardware and some requiring a working knowledge of Verilog or VHDL. The two main implementations discussed here are known as Quantum Machines [16] and Artiq [3]. Both utilize a python style language with closed hardware systems.

Artiq provides a python API in which you can program their many hardware products to perform real time calculations. Artiq effectively abstracts away the FPGA underlying the quantum processes. While the interactions between the FPGA and quantum process are hidden, the timing critical aspects are viewable and accessible to users. This ends up putting a large emphasis on real time calculations for timing critical applications.

Figure 5-1: Quantum Machines brought together a range of engineers and scientist to build their flexible platform

Quantum Machines provides a single one stop shop for quantum experiments and control with a processor capable of interfacing with qunatum systems of a wide variety. Everything from superconducting qubits to ion traps to quantum dots is supported on their OPX platform. While it does not run using Python or another popular language, the language they developed, QUA, provides a easy to learn environment with much of the same capabilities as an Artiq based piece of hardware.

5.2.2 HDL Approach

In contrast to the previous section, a HDL focused approach boasts improved performance at the cost of ease of use. More familiar to the world of FPGAs and less familiar with the world of quantum computing, a lot of these implementations are more flexible and just happen to meet the specifications required to conduct quantum experiments. Though the margin with which they achieve the required performance is a useful metric to consider. The USRP LabVIEW environment reports an approximately 50% resource utilization with the entire QSCA on chip (though not tested for functionality). This margin allows for edits and feature addition as the algorithm or use case evolves.

Figure 5-2: Innovative Integration provided an industry standard FPGA for quantum research for many years.

The first noteworthy implementation in this category would actually be the Integrative Innovations FPGA card that was used as a benchmark to compare against the USRP. The II card offered good signal quality (see Figure 5-3) and was built on a resource rich Xilinx board. The capabilities of the II were strong, but it suffered from reliability issues as well as a lack of software support for drivers. While by the end of the development process a hybrid approach was being implemented with the II card and the Simulink/HDLCoder environment, work began with the Xilinx SysGen approach alongside the II card. A compatibility issue broke this work flow and for a time the work continued by hand coding a lot of the QSCA in VHDL for the II card which, as was mentioned, is very time consuming. This was part of the reason why an alternative was sought in the USRP.

Another possible implementation platform for a project like this is the Xilinx

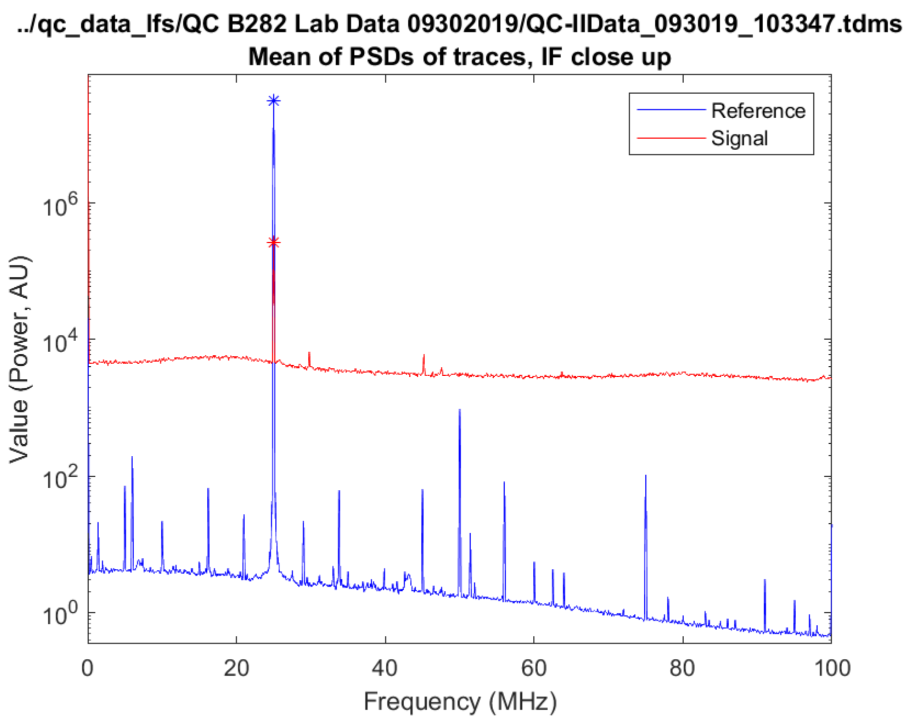


Figure 5-3: Innovative Integration provided an industry standard FPGA for quantum research for many years.

RFSoC. This FPGA platform is designed for RF signal research and fills a similar niche that the USRP does. However, it also is a Xilinx product and thus not tied to a LabVIEW development environment which provides more flexibility (or work depending on how you look at it). The frequency range specifications and signal quality benchmarks are similar or slightly better (depending on the exact model) to that of the USRP making it a likely candidate for future research. Ultimately, the optimal hardware choice for quantum sensing and control depends heavily on the precise nature of the experiment or application, but the USRP has proven to be worthy of consideration for most applications.

Chapter 6

Conclusion and Future Work

This final chapter summarizes the findings of this work by first looking back to the first byproduct of this work. This work began with a review of the quantum system that would be used as a testbed to understand how to apply sophisticated control algorithms to quantum systems. The first major product of the work is a poster seen in Figure 6-1 that laid out the initial plan and expectations as well as a general motivation and experimental process. While some of the goals were not achieved, the groundwork was laid for future USRP research to continue in the quantum sensing and control field.

6.1 Is the USRP suitable?

The USRP provides a FPGA platform that can reliably run portions of the QSCA in a similar fashion to that of the II FPGA. The moving periodogram output of the USRP and II produce comparable SNR values, overall signal quality (with the exception of additional AC coupling in the USRP), and IQ populations. Both the II and the USRP exhibit similar behavior when running the bayesian tracker which seems to indicate current stalls in the progress are more likely due to the HDLCoder development workflow. Specifically, the error can probably be traced to how HDLCoder is generating the VHDL and what settings can be altered to ensure the generated VHDL matches what is expected. One way to go about ensuring this would be to try the test bench generation feature within HDLCoder or a more ground up approach of HDL development must take place. The drawbacks of the USRP include a restricted development environment (LabVIEW) and less flexible RF hardware. Though both of these can also be viewed as strengths since the LabVIEW environment is relatively user friendly and the benefits of having RF hardware built into the USRP can come in handy. Ultimately, the USRP is a viable candidate to conduct quantum sensing and control research.

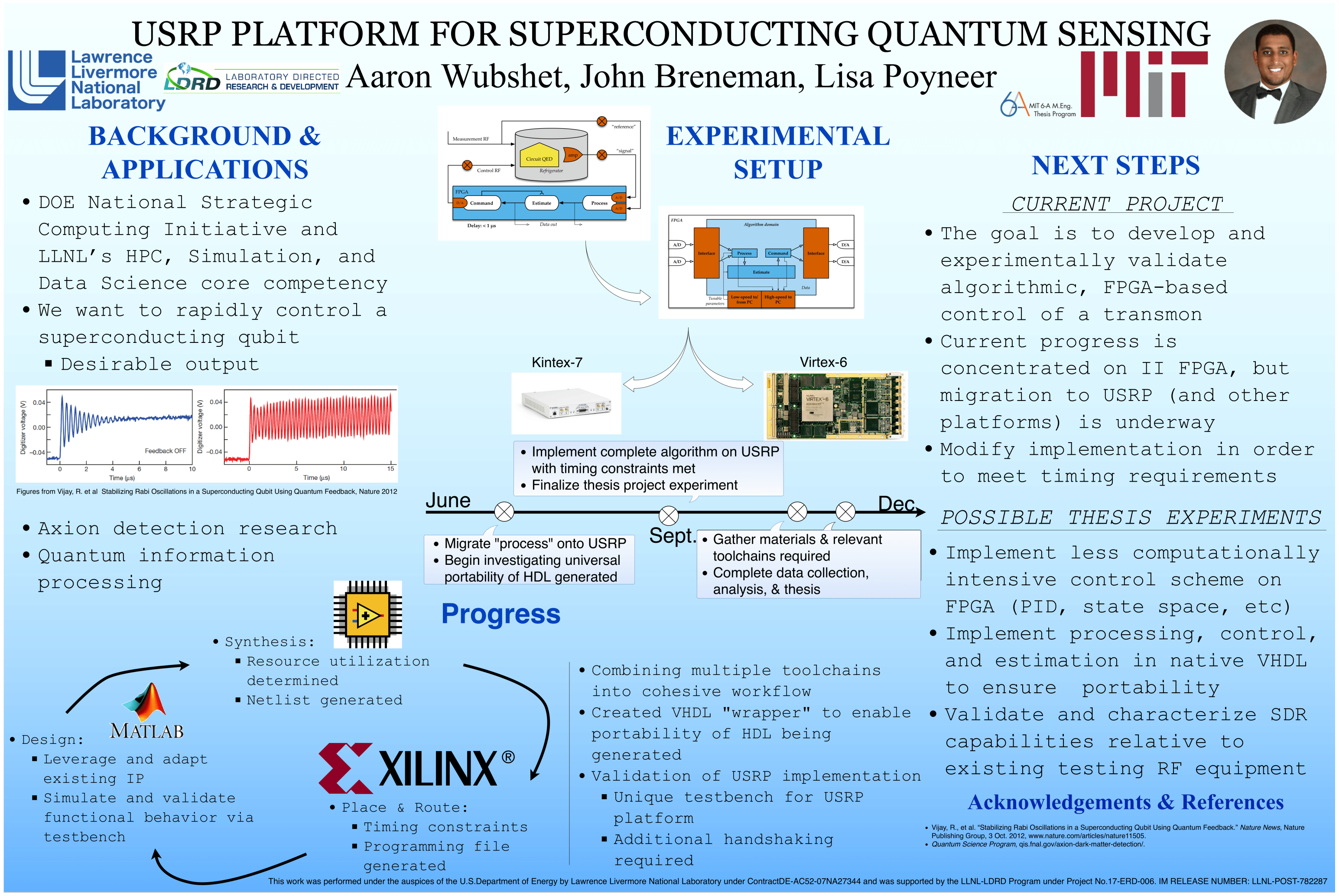
6.2 Is the QSCA complete?

This work's secondary purpose was to finish development of the QSCA's control block. An initial iteration was completed that included a PLL controller with emphasis on finding a good filter to be deployed digitally. The trade offs of rise time and filter order and filter type were explored. While no hardware testing was done with the command block, Matlab simulations indicated that in the open loop, the controller would generate command signals that theoretically would stabilize the Rabi oscillations. Unfortunately, no closed loop simulations or formal hardware testing was done, but the USRP was able to compile the full QSCA including the command block.

6.3 Future work

6.3.1 RF Capabilities

There were a few USRP RF signal characteristics that should still be investigated to further understand and solidify that the USRP is a suitable platform for continuing development of the QSCA. The first would be to determine how the USRP's signal characteristics behaved at different input frequencies. While not an immediate concern since the quantum system used was tuned to 25 MHz, the external heterodyne could have been set to any frequency in the USRP's acceptable range of 10 MHz to 6 GHz. Thus checking the SNR and PSDs of data collected by the USRP at different input frequencies would provide useful insight to the signal quality performance of the USRP. Another useful experiment would be to understand the precise reason for



USRP PLATFORM FOR SUPERCONDUCTING QUANTUM SENSING

Lawrence Livermore National Laboratory   LDRD LABORATORY DIRECTED RESEARCH & DEVELOPMENT   Aaron Wubshet, John Breneman, Lisa Poyneer   MIT 6-A M.Eng. Thesis Program

BACKGROUND & APPLICATIONS

- DOE National Strategic Computing Initiative and LLNL's HPC, Simulation, and Data Science core competency
- We want to rapidly control a superconducting qubit
  - Desirable output

Figures from Vijay, R. et al Stabilizing Rabi Oscillations in a Superconducting Qubit Using Quantum Feedback, Nature 2012

- Axion detection research
- Quantum information processing

EXPERIMENTAL SETUP

Kintex-7   Virtex-6

- Implement complete algorithm on USRP with timing constraints met
- Finalize thesis project experiment

June   Sept.   Dec.

- Migrate "process" onto USRP
- Begin investigating universal portability of HDL generated

- Gather materials & relevant toolchains required
- Complete data collection, analysis, & thesis

Progress

- Design:
  - Leverage and adapt existing IP
  - Simulate and validate functional behavior via testbench
- Synthesis:
  - Resource utilization determined
  - Netlist generated
- Place & Route:
  - Timing constraints
  - Programming file generated

- Combining multiple toolchains into cohesive workflow
- Created VHDL "wrapper" to enable portability of HDL being generated
- Validation of USRP implementation
  - Unique testbench for USRP platform
  - Additional handshaking required

NEXT STEPS

CURRENT PROJECT

- The goal is to develop and experimentally validate algorithmic, FPGA-based control of a transmon
- Current progress is concentrated on II FPGA, but migration to USRP (and other platforms) is underway
- Modify implementation in order to meet timing requirements

POSSIBLE THESIS EXPERIMENTS

- Implement less computationally intensive control scheme on FPGA (PID, state space, etc)
- Implement processing, control, and estimation in native VHDL to ensure portability
- Validate and characterize SDR capabilities relative to existing testing RF equipment

Acknowledgements & References

- Vijay, R., et al. "Stabilizing Rabi Oscillations in a Superconducting Qubit Using Quantum Feedback." *Nature News*, Nature Publishing Group, 3 Oct. 2012, www.nature.com/articles/nature11505.
- *Quantum Science Program*, qis.fnal.gov/axion-dark-matter-detection/.

This work was performed under the auspices of the U.S.Department of Energy by Lawrence Livermore National Laboratory under ContractDE-AC52-07NA27344 and was supported by the LLNL-LDRD Program under Project No.17-ERD-006. IM RELEASE NUMBER: LLNL-POST-782287

Figure 6-1: Preliminary poster describing this work

the differing shape of the noise floor when comparing PSDs of the II to that of the USRP. The low frequency trends don't match up and it would be interesting to see if that trend still held if IF was at a higher frequency like 50 MHz.

6.3.2 Closed Loop Control

Since the bayesian tracker is still in the process of being validated the command block was not implemented on either the II or the USRP. However, it was simply compiled to see if the USRP could fit (and meet timing) with the entire QSCA on fabric. This was successful but no test was done on the resulting implementation. After both the Command and Estimate stages are validated for the USRP, a closed loop control experiment could begin to see how well the qubit's Rabi oscillations could be sustained using the USRP for feedback. The validation of the command block would involve completing actual closed loop simulations before running the algorithm on the USRP.

6.3.3 Full State Feedback

After the closed loop control experiment is conducted, the next logical step would be to implement a more sophisticated control scheme in form of full state feedback on the qubit's state. Since the bayesian tracker acts as a sophisticated state estimator, a LQR or similar algorithm could be used to generate gain values for closed loop control. This would also be computationally intensive so timing and resource utilization would be a concern, but if significant enough optimizations could be made, the benefits of full state feedback over the pseudo-PLL controller currently in development would allow for easy control of many qubits with better distrubance rejection and optimal response times.

Appendix A

Tables, Figures, and Code

Power Level	SNR
260 mV	17 dB
200 mV	9.93 dB
140 mV	5.38 dB
80 mV	2.52 dB
40 mV	1.34 dB

Table A.1: Ground State SNRs

Power Level	SNR
260 mV	17.22 dB
200 mV	10.87 dB
140 mV	5.15 dB
80 mV	2.76 dB
40 mV	1.39 dB

Table A.2: Excited State SNRs

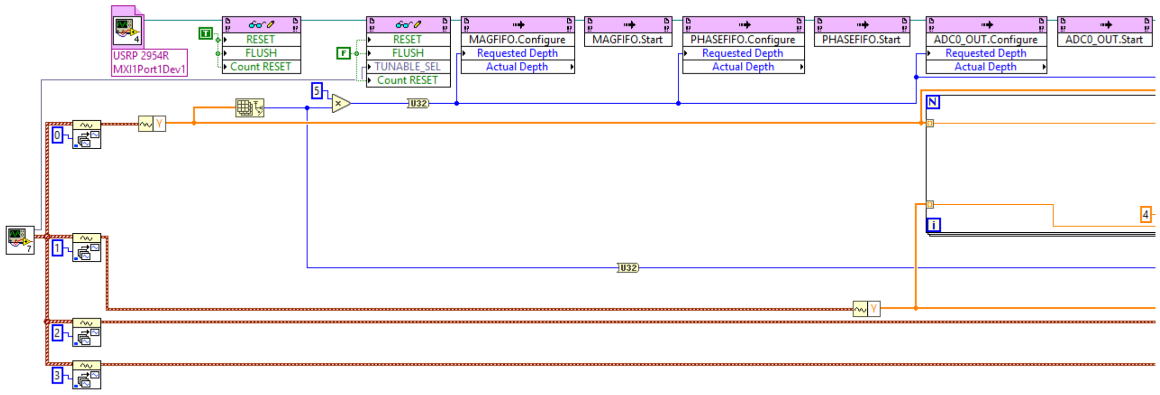


Figure A-1: Moving Periodogram Case Study Labview code

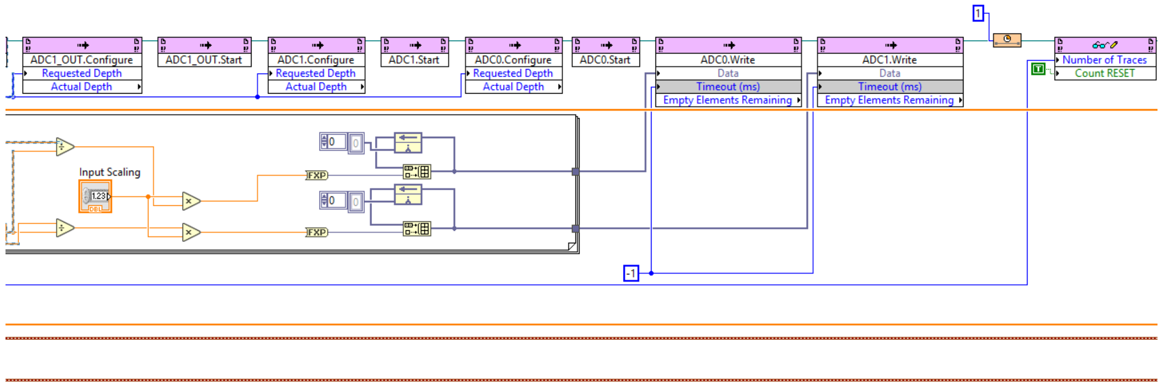


Figure A-2: Moving Periodogram Case Study Labview code

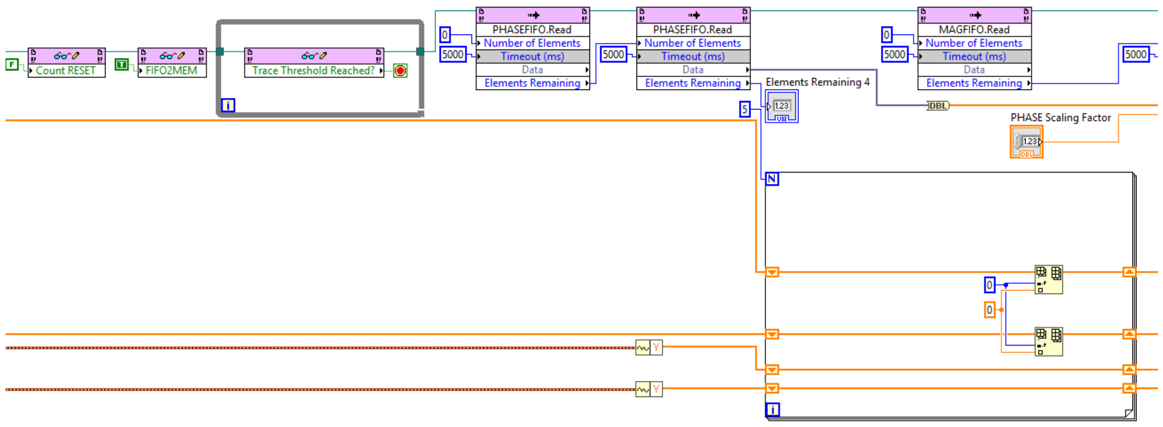


Figure A-3: Moving Periodogram Case Study Labview code

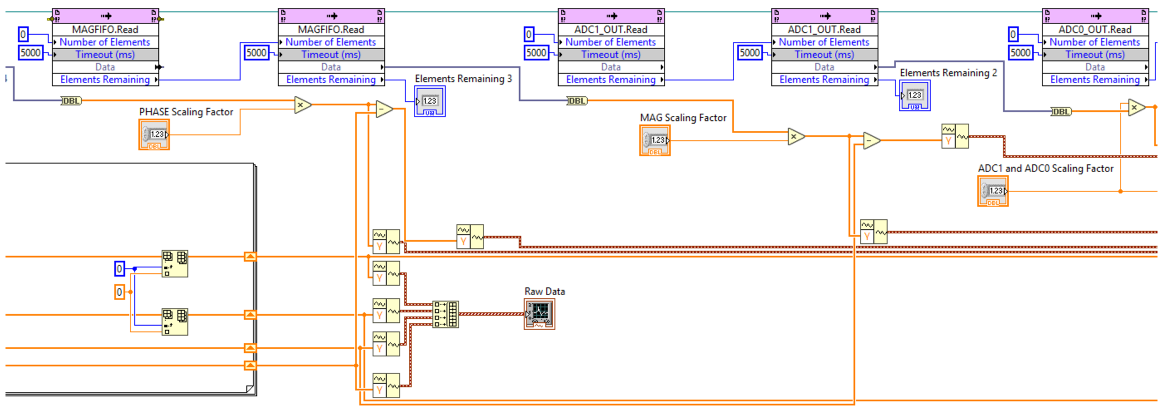


Figure A-4: Moving Periodogram Case Study Labview code

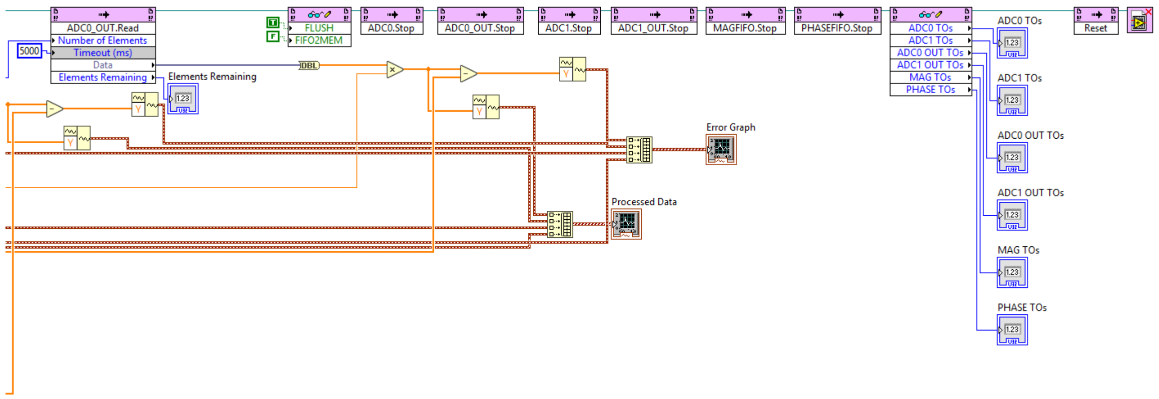


Figure A-5: Moving Periodogram Case Study Labview code

Bibliography

[1] et al. Andrew C. Doherty. Quantum feedback control and classical control theory. *Phys Rev Lett*, 1999.

[2] R. Barends, J. Kelly, A. Megrant, D. Sank, E. Jeffrey, Y. Chen, Y. Yin, B. Chiaro, J. Mutus, C. Neill, P. O'Malley, P. Roushan, J. Wenner, T. C. White, A. N. Cleland, and John M. Martinis. Coherent josephson qubit suitable for scalable quantum integrated circuits. *Phys. Rev. Lett.*, 111:080502, Aug 2013.

[3] Sébastien Bourdeauducq. Artiq. M-Labs, 2007.

[4] S. A. Caldwell, N. Didier, C. A. Ryan, E. A. Sete, A. Hudson, P. Karalekas, R. Manenti, M. P. da Silva, R. Sinclair, E. Acala, N. Alidoust, J. Angeles, A. Bestwick, M. Block, B. Bloom, A. Bradley, C. Bui, L. Capelluto, R. Chilcott, J. Cordova, G. Crossman, M. Curtis, S. Deshpande, T. El Bouayadi, D. Girshovich, S. Hong, K. Kuang, M. Lenihan, T. Manning, A. Marchenkov, J. Marshall, R. Maydra, Y. Mohan, W. O'Brien, C. Osborn, J. Otterbach, A. Papageorge, J.-P. Paquette, M. Pelstring, A. Polloreno, G. Prawiroatmodjo, V. Rawat, M. Reagor, R. Renzas, N. Rubin, D. Russell, M. Rust, D. Scarabelli, M. Scheer, M. Selvanayagam, R. Smith, A. Staley, M. Suska, N. Tezak, D. C. Thompson, T.-W. To, M. Vahidpour, N. Vodrahalli, T. Whyland, K. Yadav, W. Zeng, and C. Rigetti. Parametrically activated entangling gates using transmon qubits. *Phys. Rev. Applied*, 10:034050, Sep 2018.

[5] Wei Cui. Feedback control of rabi oscillations in circuit qed. *Physics Rev A*, 2013.

[6] Mohammed El-Hajjar. Demonstrating the practical challenges of wireless communications using usrp. *IEEE*, 2014.

[7] V.S. Shumeiko G. Wendin. Superconducting quantum circuits, qubits and computing. *G-Animal's Journal*, 2005.

[8] Daniel Kastner. Embedded dsp: Introduction to digital filters. AbsInt, December 2002.

[9] D. Leibfried, R. Blatt, C. Monroe, and D. Wineland. Quantum dynamics of single trapped ions. *Rev. Mod. Phys.*, 75:281–324, Mar 2003.

[10] Y. Liu. Comparing and combining measurement-based and driven-dissipative entanglement stabilization. 2018.

[11] Chao-Yang Lu, Daniel E. Browne, Tao Yang, and Jian-Wei Pan. Demonstration of a compiled version of shor's quantum factoring algorithm using photonic qubits. *Phys. Rev. Lett.*, 99:250504, Dec 2007.

[12] Thomas Mannel. Theory and phenomenology of cp violation. *Nuclear Physics B*, 2007.

[13] J.M. Overduin. Dark matter and background light. *PHysics ReportsNuclear Physics B*, 2004.

[14] D. Ristè. Feedback control of a solid-state qubit using high-fidelity projective measurement. *Physics Rev Lett*, 2012.

[15] Leslie Rosenberg. The axion dark matter experiment. ADMX, 1995.

[16] ITAMAR SIVAN. Opx. Quantum Machines, January 2020.

[17] Kate Streit. Better than star wars: Chemistry discovery yields 3-d table-top objects crafted from light. SimpleMost, October 2017.

[18] Kate Streit. Hover cars could be a reality sooner than we think. SimpleMost, October 2017.

[19] L. et al. Vandersypen. Experimental realization of shor's quantum factoring algorithm using nuclear magnetic resonance. *Nature*, 2001.

[20] R. Vijay. Quantum feedback control of a superconducting qubit: Persistent rabi oscillations. *Nature*, 2012.