The University of Colorado Boulder

Colorado Nanosat Atomic Clock Testbed

Fall 2020 Semester Report

Authors
Luciana Schement
Quinn LaBarge
Prayag Desai
Caroline Dixon
Henry Dixon
Daniel Dowd
Christopher Flood

Sponsor POC
Dr. Joanna Hinks
Advisor
Dr. Penina Axelrad

December 12, 2020

Contents

List of Figures

List of Tables

1 Introduction

In August 2019, the DoD publicly released an unclassified version of its "Strategy for the Department of Defense Position, Navigation, and Timing (PNT) Enterprise: Ensuring a U.S. Military PNT Advantage." Emphasizing the foundational role of GPS PNT in Joint Force operations, the Strategy warns that "space-based PNT services provided by GPS will be targeted and will not always be available in contested military operating areas. . . complementary PNT capabilities must be applied." [1]

While GPS is best known as a navigation system, one of its critical functionalities in providing global PNT capability is disseminating precise time, time intervals, and frequency. Communications systems use both time and frequency to maintain carrier frequencies and data-bit phase timing, and military networks have especially stringent timing requirements for the synchronization of data encryption and decryption equipment [4]. Since the 1950s, the gold standard for timekeeping has been ground-based atomic clocks which are generally not suitable for spaceflight or ground vehicles due to their size, weight, power requirements, and environmental sensitivity [7]. Through its precise time and frequency dissemination capability, GPS provides a consistent, accurate, and inexpensive external frequency source to both military and civil users around the globe.

As a part of the "complementary PNT capabilities" aimed at mitigating the inherent vulnerability of DoD dependence on GPS, the Air Force Research Lab (AFRL) is interested in developing accurate onboard timing technologies that are less reliant on frequent GPS time corrections. The AFRL Space Vehicles Directorate is sponsoring a graduate student project under Dr. Penina Axelrad at the University of Colorado Boulder in alignment with the AFRL mission to develop and transition emerging technologies for military space applications. The Colorado Nanosat Atomic Clock Testbed (CONTACT) project objective is to develop a testbed capable of ensembling three or more low-SWaP (size, weight, and power) atomic clocks and producing an accurate and robust time signal for small satellite applications. Advanced PNT satellites could use this technology to achieve a highly accurate time scale when GPS is unavailable. Satellites in GPS-denied environments with strict position accuracy requirements could also use an improved onboard time reference to support precise orbit determination.

2 Scope

2.1 Project Description

The continued development of accurate, robust onboard timekeeping technologies is foundational to the improvement of non-GPS PNT capabilities for a multiplicity of military and civil applications. Complementary PNT enabled by low-SWaP (Size, Weight, and Power) onboard timing systems can increase the resilience of communication and navigation systems in degraded or denied GPS environments. The limited performance inherent in low-SWaP clocks can be mitigated by ensembling several independent devices and generating a steered output frequency signal based on the weighted averages of the inputs. A testbed can be used to evaluate the performance of various clock configurations and signal steering techniques as well as to measure the effects of environmental factors and clock errors. The objective of this report is to describe the design and implementation of this clock ensembling testbed on a Software Defined Radio (SDR) hardware platform during the Fall 2020 semester. Results from this work could be used in the development of low-SWaP onboard timing systems capable of meeting Precise Time and Frequency (PTF) requirements in the absence of GPS.

The testbed will be a standalone, modular, fully integrated system which takes three low-SWaP atomic clocks under test and a reference signal as inputs and generates a steered output signal along with clock characterization reports. The system will be implemented on a 4-input Software Defined Radio and programmed with the GNU Radio open-source software development toolkit. The testbed uses well-defined SDR oscillator metrology techniques to measure high-frequency signals while maintaining low onboard storage requirements. The testbed is designed to take phase and frequency measurements of signals in the 5 MHz – 200 MHz range to ensemble clocks of dissimilar accuracy and stability. The system also will also have the capability to introduce clock errors such as jumps in phase and frequency and evaluate the robustness and resilience of the steered output signal. The measured time series data (phase and frequency) are processed to produce Allan Deviations (ADEVs) and plots for the characterization of performance. A filtering subsystem estimates the bias and drift of each clock under test. The filter uses a standard clock model and the calculated variance values for filter clock state propagation. The filter outputs deviation and covariance time histories for comparison of actual errors with three-sigma error statistics. The clock estimates are also used to produce a composite clock timescale in a weighted-average ensembling algorithm. The output of the ensemble are phase and frequency tuning corrections, which are passed into the signal generation subsystem along with the corresponding reference signal to be steered. The primary functional components of the signal generation subsystem are the frequency multiplier, direct digital synthesis (DDS), and a smoothing phase-locked loop (PLL). The

frequency multiplier passes the reference input signal through a full bridge rectifier, bandpass filter, and amplifier to produce the frequency multiplied signal to clock the DDS. The DDS is the core of the signal generation subsystem and generates a steered sinusoidal signal from the phase and frequency tuning inputs. The steered digital output is passed through a PLL to generate an analog output signal that has been steered and smoothed. This output signal can be measured for controller performance analysis and signal characterization under variable testing parameters.

2.2 Project Objective

The Colorado Nanosat Atomic Clock Testbed (CONTACT) three-year project objective is to design, build, and operate a clock ensemble testbed for low cost, high accuracy timing applications. The function of the testbed is to measure multiple Chip-Scale Atomic Clock (CSAC) inputs, combine them using filtering and ensembling algorithms, and generate a steered output signal more accurate and robust than any one of the individual clocks. The testbed system will be modular, standalone, and support customization including multiple clock inputs, ensembling methods, and fault management capabilities.

2.2.1 Semester Objective

The CONTACT project is heading in to its fourth semester in Fall 2020. The Ettus N310 software defined radio (SDR) has been selected for the clock ensemble testbed. Work on the testbed this Fall focused on implementing the measurement, filtering and ensembling, and fault injection components of the signal processing chain in GNU radio. The steering system used to realize the clock ensemble was validated prior to development in GNU radio. In addition, hardware tests were performed to characterize the stability of signals input to the SDR.

The MAXWELL CubeSat is nearing its launch date in 2021. The CSAC Experiment, designed by the CONTACT team in cooperation with the MAXWELL ADCS team, entered its preliminary testing phase this Fall. Efforts this semester centered around defining and executing RF simulator tests, thermal chamber tests, and MAXWELL ADCS subsystem interfacing tests. These objectives will be discussed in detail in subsequent sections.

2.3 Resources

2.3.1 Roles and Responsibilities

This semester report is part of a structured reporting plan which also includes internal monthly individual progress reports and formal reviews with AFRL. This reporting plan is intended to mitigate the effects of team discontinuity between academic years or between semesters as well as facilitate stakeholder involvement in the project development. Shown below is the CONTACT team organization chart for Fall 2020 as well as a description of roles and responsibilities for select stakeholders. Project POC: luciana.schement@colorado.edu

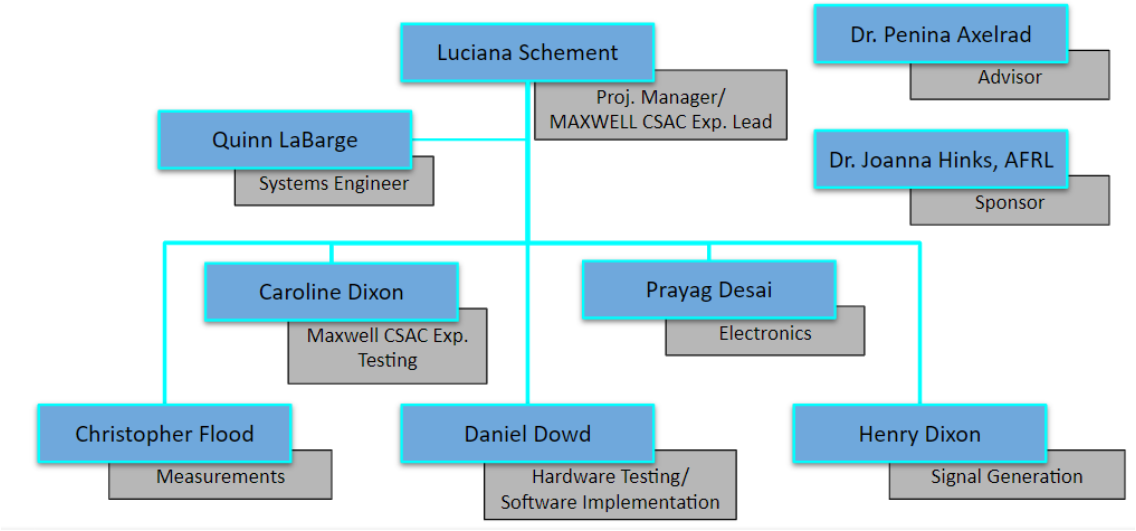


Figure 1: Fall 2020 Org Chart

Members	Role/Description
Dr. Joanna Hinks	Sponsor POC - provides requirements, use cases, and guidance on project development. Reviews progress at scheduled intervals.
Dr. Penina Axelrad	Faculty advisor – provides subject area expertise and direction to other useful university resources and personnel
Luciana Schement	Project Management, MAXWELL CSAC experiment lead
Quinn LaBarge	Systems Engineer, SDR software subsystem development
Prayag Desai	CSAC breakout board development
Caroline Dixon	MAXWELL CSAC experiment testing and integration
Henry Dixon	Signal generation lead, SDR hardware and software testing
Daniel Dowd	Filtering and ensembling subsystem lead, SDR hardware testing
Christopher Flood	Measurement subsystem lead, software subsystem interfacing

Table 1: Roles and Responsibilities

2.3.2 Other Resources

Resource	Point of Contact	Reason for Need
Aerospace Instrumentation Shop	Trudy Schwartz	Signal measuring/analysis equipment
MATLAB/Simulink	OIT	Implementation of Kalman filter and signal generator
COMPASS Lab	Dr. Penina Axelrad	GPS receiver and antenna, electronics workstation, and RF simulator
NIST Instrumentation	Dr. Stefania Römisch	Clock measurement interface for development of Kalman filter and signal generator
Microsemi	Richard Foster/ Peter Cash	Contacts at the company that produces the CSACs
RF Simulator	Dr. Dennis Akos	CSAC board design specifications, CSAC communication, and RF simulation setup
MAXWELL Graduate Project	Anastasia Muszynski	MAXWELL contacts to ease integration of the CSAC experiment

Table 2: Resource List

2.4 Schedule

The project team has completed documentation deliverables and schedule-driven reviews concluding with an end-of-semester review on December 10, 2020. Efforts during the Fall 2020 semester have included the following:

- Simplified experiments performed with LF signals to gain fundamental understanding of differences between steering methods
- Used the Ettus N310 SDR to bring 10 MHz clock signals to lower beat frequency (100 Hz)
- Implemented Kalman filter in GNU Radio Companion
- Fault injection system successfully inserts frequency and phase jumps onto a signal
- Design, assembly, and test of CSAC breakout board
- CSAC experiment ground testing development using Spirent STR4500 RF simulator

Moving forward into Spring 2021, development efforts will include:

- Perform a trade study to determine the best method for making phase measurements
- Conduct longer-duration live tests with multiple CSAC inputs
- Conduct the high rate and baseband steering algorithm experiments with N200/210 SDR
- Create a plan for a fault attribution system to isolate errors in each CSAC
- Conduct thermal chamber tests using updated thermal profile from MAXWELL

2.5 Budget

The budget spreadsheet shown in Table 3 shows an accounting for the first two years of project funds which extends until the end of CY20. Included in this spreadsheet are all previous expenses and all planned major expenses for CY20.

Source/Expense Item	Project Funds Available	Expected Expenses
AFRL Grant (less CU overhead)	$48,000	
Previous Expenditures		$30,361
Fall 2020 Expenditures		
Department Overhead		$0
Clocks		$0
Electronics		$0
Software		$0
Subtotal	$48,000	$31,216
Final Balance	$16,784	

Table 3: Budget

3 System Description

3.1 Requirements

1. The measurement subsystem shall be capable of measuring the phase of selected clocks against a reference clock

    1.1. The measurements shall be taken at a rate of no less than once per minute

    - Rationale: To enable optimal filtering
    - V&V: Timestamp data verifies sampling rate

    1.2. The measurement system shall store up to 24 hours of continuous measurements

    - Rationale: Upper bound for potential continuous runtime
    - V&V: Timestamp data verified duration of measurement

    1.3. The measurements shall be outputted to the filtering system within TBD minutes of being generated

    - Rationale: To enable optimal filtering and steering
    - V&V: Timestamp data can be used to find lag time

    1.4. The measurement system error shall be less than the specified performance error of the highest quality clock

- Rationale: Necessary for proving the testbed more effective than a single clock
- V&V: Analysis to identify potential biases

1.5. The measurement system shall be capable of measuring at least 3 x 10 MHz oscillators

- Rationale: Required for ensemble
- V&V: Demonstrate simultaneous measurement on multiple input channels

1.6. The measurement system shall be capable of utilizing an external reference clock

- Rationale: Necessary for measuring input clocks
- V&V: Analysis of measurement subsystem architecture

2. The testbed shall be capable of characterizing oscillators

2.1. The characterization system shall accept clock phase as an input

- Rationale: generic input for ease of use by different subsystems
- V&V: Analysis of characterization subsystem architecture

2.2. The characterization system shall generate Allan Deviations

- Rationale: useful clock characterization tool
- V&V: Analysis using Stable32

2.3. The characterization system shall calculate and output the clock variance to the filtering system

- Rationale: required for proper filter subsystem operation
- V&V: produce correct calculations for clocks of known variances

3. The testbed shall estimate clock biases and drifts using one or more filtering approaches

3.1. The filtering system shall produce deviation estimates of each clock relative to a chosen reference clock

- Rationale: required for ensembling algorithm
- V&V: estimation errors meet the 3-sigma error statistics

3.2. The filtering system shall consist of generic modular functions

- Rationale: to easily change between filtering algorithms
- V&V: Demonstrate multiple filter modules

3.3. The filtering system shall be capable of processing output data from measurement system

- Rationale: prove proper interface between subsystems
- V&V: prove automatic ability of filtering subsystem to take in and process data directly from measurement subsystem

4. The testbed shall be capable of ensembling filtering results

4.1. The ensembling algorithm shall take in the state estimates from the filtering system for each clock

- Rationale: prove proper interface between subsystems
- V&V: prove automatic ability of ensembling subsystem to take in and process data directly from filtering subsystem

4.2. The ensembling algorithm shall output corrections based on the clock estimate to the signal generation system at TBD rate

- Rationale: required for signal generation algorithm
- V&V: prove autonomous ability of ensembling subsystem to feed data to the signal generation subsystem

5. The testbed shall produce a steered output signal

5.1. The signal generation system shall accept as input a 10 MHz reference clock signal

- Rationale: signal to be steered by system
- V&V: measure analog input signal with oscilloscope probe/LabVIEW waveform monitor

5.2. The signal generation system shall generate a steered 10 MHz sinusoidal output signal

- Rationale: overall objective of the testbed
- V&V: evaluation of output signal through feedback to measurement subsystem

5.3. The signal generation system shall feed back the steered output signal to the measurement system

- Rationale: characterization of the steered output signal generated
- V&V: prove functionality of feedback routing to measurement subsystem

5.4. The signal generation system shall be capable of processing the clock ensemble state that is output from the ensembling algorithm at a TBD rate

- Rationale: Generate a sufficiently accurate and robust output signal
- V&V: Analysis of signal generation system design and time tagging of the control inputs

6. The system shall be capable of testing the effects of faults in the clock ensemble

6.1. The system shall be capable of injecting faults through the modification of measurement data

- Rationale: Determine robustness of system and resilience to errors in measurements acquisition
- V&V: Demonstrate fault injection capability

6.2. The system shall be capable of inducing faults by varying environmental conditions

- Rationale: Explore environmental effects on system performance
- V&V: Demonstrate environmental controls and impact on output signal

3.2 ConOps

The Concept of Operations for this project in Figure 2 shows the nominal system use case.

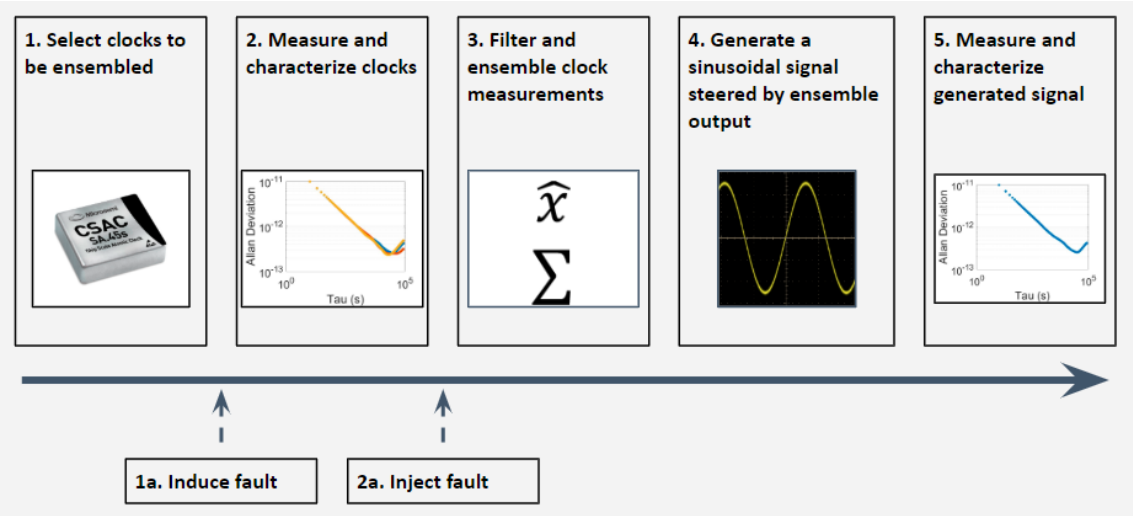


Figure 2: Concept of Operations

Moving from left to right along the ConOps diagram, the nominal use case includes:

1. Clock selection – identify low-SWaP atomic clocks for test
2. Measurement – clock signals are input into the testbed for phase and frequency measurement with respect to a reference clock
3. Fault induction/injection – the user can evaluate system performance and resilience under non-ideal conditions by inducing or simulating clock errors
4. Filter & Ensemble – measured clock signals are filtered to provide estimates of the clock states with respect to the implicit ensemble mean (IEM) - the filter outputs phase and frequency offsets to be used for signal generation
5. Steered signal output – an oscillator is steered to a desired phase and frequency based on the ensembled signals (this is the overall output of the timing system)
6. Characterization - describe the stability performance of the generated signal over various time scales

3.3 Functional Block Diagram

The functional block diagram Figure 3 shows the boundary of the standalone testbed system. The full integrated testbed will be implemented on the Ettus N310 controlled through GNU Radio on a connected PC. The testbed will accept clock signal inputs and generate a steered output signal along with characterization reports. Each subsystem block within the testbed is described below. Subsequent sections of this report cover each of these subsystems in detail, as well as overall system hardware implementation and the MAXWELL CSAC experiment. Each section details past work, development completed this semester, and next steps.

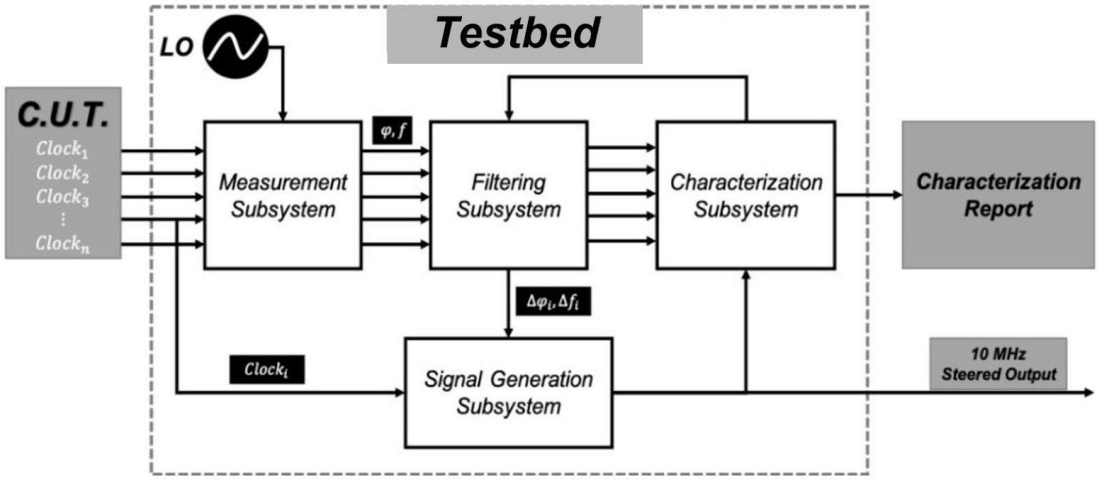


Figure 3: Functional Block Diagram

3.3.1 Measurement Subsystem

The measurement subsystem accepts input signals from multiple clocks and makes phase and frequency measurements for comparison between the selected clocks. Ideally, the measurement subsystem will be capable of accepting multiple types of clocks for simultaneous input. A time interval counter, frequency counter, or phase comparator allows for simultaneous comparison of each input signal to a common reference clock. The measurement data are formatted and sent to the characterization and filtering subsystems.

3.3.2 Characterization Subsystem

The characterization subsystem processes clock time series data (phase and frequency) to produce an Allan Deviation (ADEV) calculation and plot. The ADEV is used to calculate the variances of the noise sources making up the clock signals. The variances are sent to the filter and ensemble subsystem. The system also outputs phase and frequency ADEV plots including theoretical clock reference lines (from the supplier data sheet) and error bars.

3.3.3 Filter/Ensemble Subsystem

The filtering subsystem produces estimates for the bias and drift of each clock. The filter uses a standard clock model and the variance values from the characterization subsystem for the filter clock state propagation. The differenced clock measurements from the measurement subsystem are used as input to the filter observation update. The filter outputs deviation and covariance time histories to show agreement with three-sigma error statistics. The clock estimates are fed to the ensembling subsystem. The ensembling subsystem produces a composite clock timescale. The clock time history estimates, from the filtering subsystem, are input into a weighted average ensembling average algorithm. The clock ensemble is fed to the signal generation subsystem.

3.3.4 Signal Generation Subsystem

The signal generation subsystem serves the role of producing the realization of the implicit ensemble mean. This is accomplished by steering the frequency of one of the clocks under test towards the IEM. The fundamental concepts used to form the nominal design for this subsystem are based off of a method which some atomic frequency standards, such as hydrogen masers, use to generate analog timing signals [5]. In this method, a stable frequency source is used as the timing source for a frequency synthesis operation. The synthesized output is used to drive a phase-locked loop (PLL) with a voltage controlled crystal oscillator (VCXO); where the VCXO produces the clean analog timing signal.

This approach is applied to signal generation for the clock ensemble testbed as follows. One of the ensembled 10 MHz CSAC signals is selected as the steering reference ($CSAC_i$) to serve as the stable frequency source. The steering reference is frequency multiplied by a factor of four, and provides a timing source to a direct digital frequency synthesis (DDS) routine. The DDS behaves as a microphase-stepper and applies the Δf steering adjustments to the nominal 10 MHz frequency source. Optimal Δf steering adjustments are computed by a state estimator and regulator that are updated by phase error estimates ($\Delta\hat{\phi}$), with respect to the IEM, received from the ensemble subsystem Kalman Filter. Since this synthesis routine is clocked at a rate of four times the nominal output frequency, it produces a low resolution sinusoidal output containing much unwanted harmonic noise. The low resolution steered signal is smoothed by driving a PLL, and the output of the voltage controlled oscillator in the PLL becomes the steered and smoothed timing signal output. A functional block diagram for the signal generation subsystem is provided in Figure 4 below.

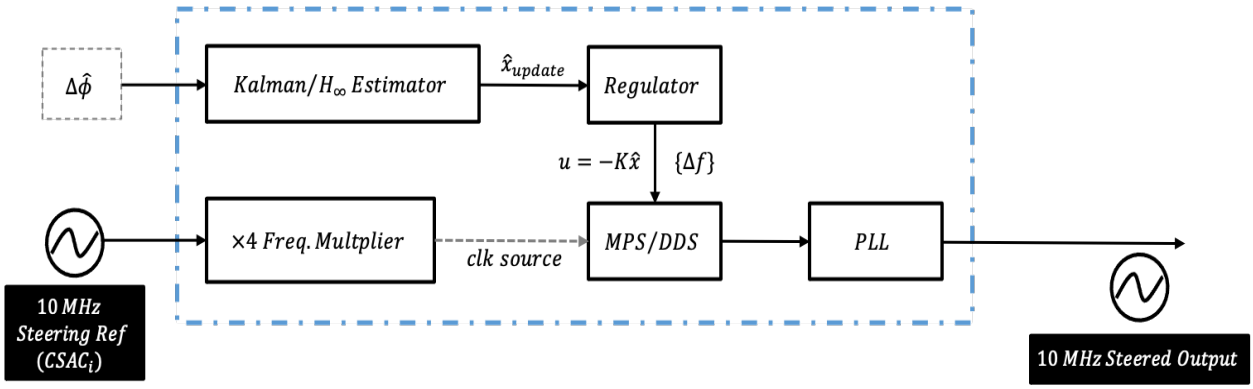


Figure 4: Functional block diagram for the nominal signal generation subsystem design.

4 CSAC Experiment

The CONTACT team is developing a clock experiment to be flown onboard the UNP-9 MAXWELL mission. The primary purpose of this experiment is to assess the accuracy of a Chip-Scale Atomic Clock (CSAC) on a space platform. This CSAC is used as an external clock with a NovAtel OEM 729 GPS receiver. The clock-receiver setup collects data that will characterize the CSAC's performance in space. The MAXWELL Attitude Determination and Control System (ADCS) records the position, clock bias, bias rate, temperature, pseudorange and phase information from the experiment. The clock bias information is used to generate Allan Deviation plots, whereas the temperature information allows the development of a relationship of the variation in temperature to the CSAC performance. The pseudorange and phase information allows for improved accuracy in the bias estimation.

The primary work done in Fall 2020 included the ground testing and space preparation for this experiment. The work started with unlocking the MAXWELL engineering development unit (EDU) NovAtel receiver to be used in space mode. Testing on the NovAtel EDU receiver was conducted using a Spirent RF simulator and preparation for thermal chamber testing was initiated. Along with this, an abstract was submitted and accepted for the Precise Time and Time Interval Systems and Applications conference. The next subsections will elaborate on this work in detail.

4.1 Ground Testing

Ground testing was a focus for the MAXWELL team this semester as they are preparing to deliver hardware in 2021. For the CSAC experiment, there are two main reasons for ground testing the receiver-clock setup. First, ground testing ensures that the receiver and clock interface properly. Second, we can characterize the performance of the CSAC on the ground by replicating space-like scenarios. This yields a baseline for the CSAC performance that the team can use to compare to the onboard performance in space. To replicate the space environment, we will conduct RF simulator and thermal chamber tests.

In order to perform these tests, the EDU receiver needed to be unlocked to be used in space mode. The receiver can produce a position solution when the speed that the receiver is moving is below a certain threshold. If the receiver is moving at a speed that is above the threshold, the receiver will not be able to get a lock on the signal. However, the receiver can be "unlocked" which allows the receiver to process signals at speeds greater than the threshold. When the receiver is on orbit, it will be traveling faster than the threshold, so we need to unlock the EDU receiver before we can run a space simulation.

4.1.1 RF Simulator Testing

For the RF simulator test, a Spirent STR4500 simulator was connected to the MAXWELL EDU NovAtel OEM729 GPS receiver through the antenna port on the NovAtel development board. Additionally, the CSAC was connected to the receiver as an unsteered external oscillator via the development board. The test setup is shown in Figure 5. The STR4500 sends simulated signals to make the receiver think that it is in a specific scenario. For example, there are simulations that replicate the motion of a car, airplane, boat, or satellite. For ground testing the clock-receiver setup, we want to use a satellite in LEO simulation as the MAXWELL experiment will be flown in LEO.

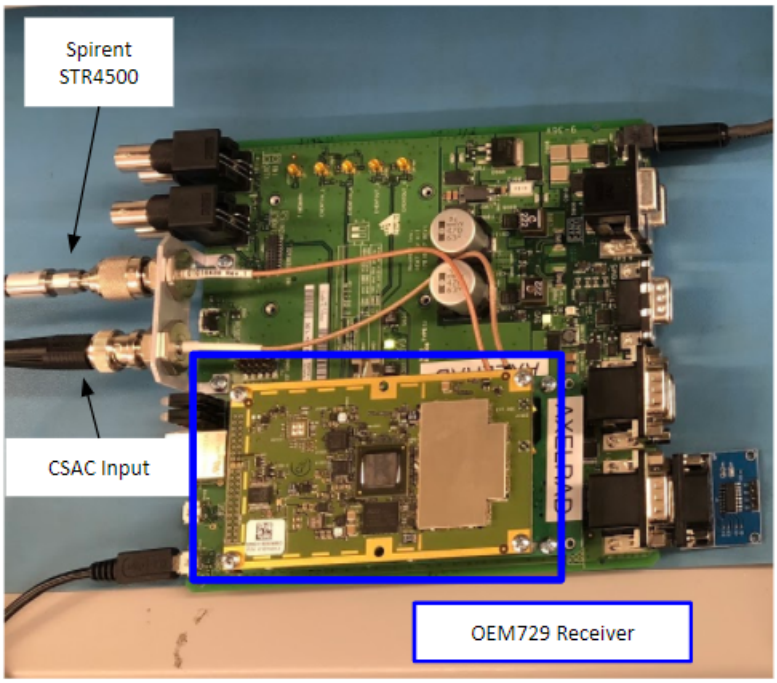


Figure 5: RF Simulator Setup

The simulation that we chose was a circular orbit in LEO. It has a run time of four hours and we collected pseudorange, phase, bias, bias rate, and positioning messages from the EDU receiver at a sampling rate of 1 Hz. The simulation must start running before the development board and receiver are turned on. This order was deemed necessary to get the experiment to run successfully. This means that we are not able to collect data for the full 4 hours. Rather, the total time that data is collected is approximately 3.5 hours.

Initially, the team hoped to use the RF simulator tests to characterize the performance of the CSAC by creating Allan Deviation plots. However, after running some initial RF simulator tests, we discovered that the clock bias that was given in the EDU receiver messages represented the clock bias of the RF simulator, not the CSAC. This is because the STR4500's internal clock is not as accurate as the CSAC, so the clock bias in the simulator is greater than the CSAC. In order for the receiver clock messages to represent the CSAC, we will need to do a locked clock test. In a locked clock test, the receiver and the Spirent simulator would both use the CSAC as an external clock. The team plans to investigate the STR4500 to assess the feasibility of conducting this test.

4.1.2 Preliminary Tests with STR4500

Before completing the LEO simulation, preliminary tests with the Spirent simulator were conducted to understand how to setup the simulator. Two simulations were used to run these

preliminary tests: a boat scenario and a car scenario. In the boat scenario, the receiver is moving at a velocity of 5 m/s. While in the car scenario, the receiver has a velocity of 70 MPH.

Figure 6 shows skyplots from the receiver and simulator computer applications for the boat simulation. The left skyplot shows the GPS satellites that the STR4500 has simulated. The picture on the right shows the GPS satellites that are in view for the receiver. By comparing these two pictures, we can see that the skyplots are identical and verify that the receiver is properly acquiring signals from the simulator.

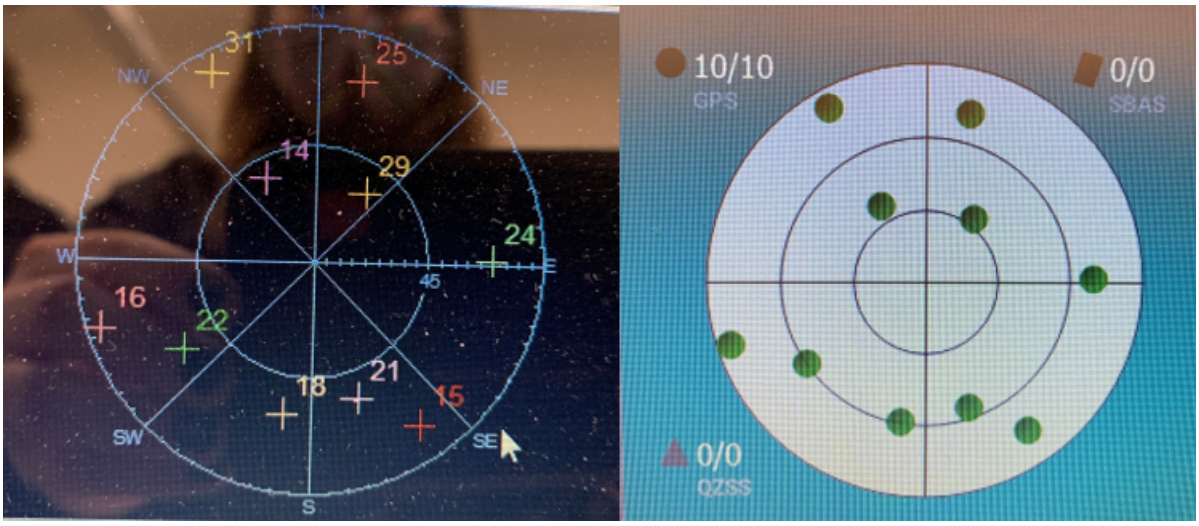
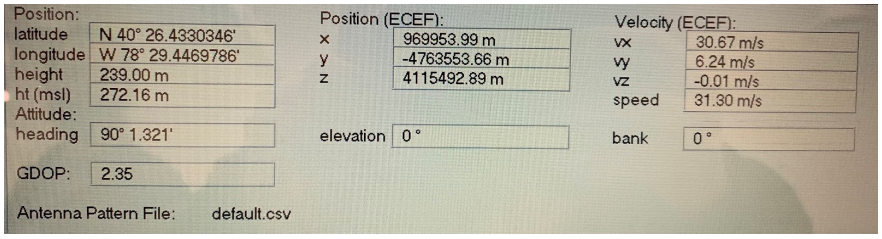
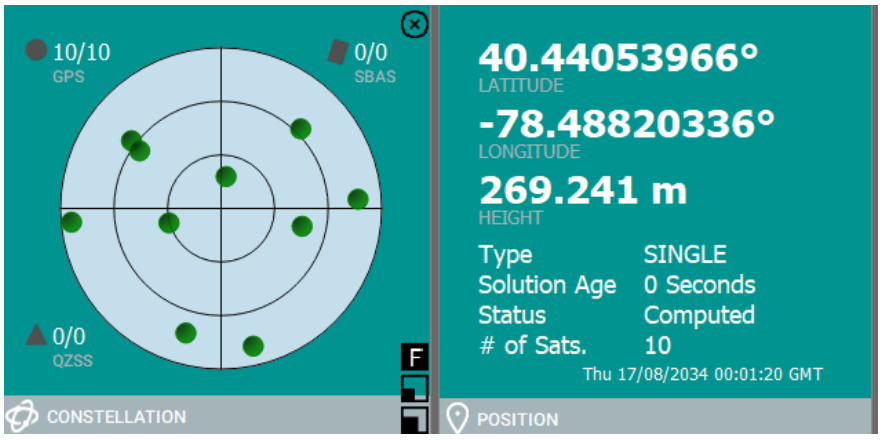


Figure 6: Skyplots from Spirent simulator software and NovAtel Connect

Using the car scenario, we tested the receiver's ability to accurately calculate its position. Figure 7 shows the position and velocity data from the Spirent simulator software and the latitude, longitude, and height from the NovAtel Connect software. Looking at this figure, it can be determined that the receiver accurately calculated the latitude to $1x10^{-5}$ degrees and the longitude to $1x10^{-2}$ degrees. This verifies that the receiver can accurately calculate its position using the simulated signals from the STR4500.



(a) Spirent position and velocity data



(b) NovAtel latitude, longitude, and skyplot

Figure 7: Car scenario Spirent and NovAtel positioning information

4.1.3 Thermal Chamber Testing Preparation

An important part of the CSAC experiment is determining how temperature affects the performance of the clock. We can conduct testing on the ground with a thermal chamber to get a better understanding of this relationship. In the Spring of 2020, the MAXWELL team purchased a thermal chamber. However, due to COVID-19 access to the lab and the thermal chamber was restricted this Fall. In preparation for thermal testing, we have received training on using the thermal chamber from the MAXWELL testing lead and written the first draft of a procedure.

4.1.4 Phase Offset Test

The CSAC has the ability to measure the phase difference between its own 1PPS signal and an external 1PPS signal. In this test, the phase difference between the CSAC and the Rubidium atomic clock was measured for 12 hours. Although currently not completed, the eventual goal is to compare this method of measuring phase differences with the methods outlined in the Measurement Subsystem section. The initial plan was to utilize the CSACdemo software; however, it proved to not be flexible enough to perform the required tasks. The key problem with using the CSACdemo software was that there was no way to change the

mode register. This function was essential because it was necessary to have the CSAC in mode register 0x0004 which signifies that the phase measurements will be recorded; however, the steering will be turned off. The steering capability of the CSAC enables it to steer the frequency of its own signal to that of an external oscillator. With this enabled, the phase offset essentially goes to zero since the CSAC's frequency is being steered to match the frequency of the external oscillator. Essentially, using CSACdemo, either both steering and phase measurement could be turned on, or both could be turned off, but not one or the other. It was necessary to directly communicate with the CSAC by setting up a serial port connection and essentially replicating the functionality of the CSACdemo software in a more flexible format that could provide useful data.

The CSAC communicates with non binary ASCII characters via an RS-232 connection set to 57600 8-N-1. MATLAB's serial port function was used to initialize a connection with the CSAC from a computer. It was also used to send the CSAC specific commands to the CSAC. For example, the command !6 returns the telemetry headers which include, but are not limited to Status, Alarm, Temperature, Phase, and TOD. Additionally, the command !ˆreturns the data readings of each field of the telemetry headers. A script was written that would save the telemetry data once every 10 seconds and this test ran for 12 hours. A simple time series of the collected phase offset and steer values are shown in the plot below. Note that the steer value remains at 0 the duration of the test, as desired. The positive linear nature of the phase offset is what was expected.

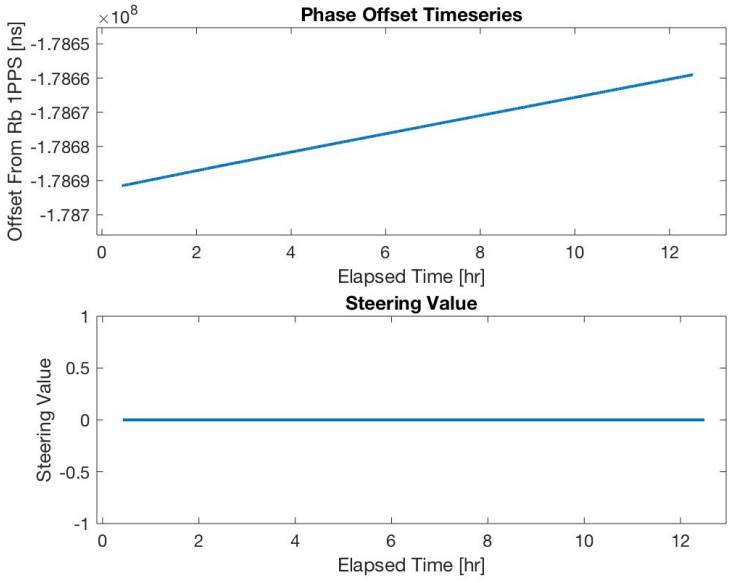


Figure 8: CSAC measured Phase and Steering time series.

4.2 Future Work

In the Winter and Spring of 2021, progress will be made on researching the locked clock testing with the RF simulator. The CONTACT team will work with the MAXWELL thermal subsystem team to conduct thermal chamber testing using an updated thermal profile. Additionally, we will be building upon the serial port connection with the CSAC and extending this to the ADCS flight software to access the CSAC data onboard the cubesat. Lastly, we plan to work the MAXWELL ADCS software developers to implement a nadir-pointing mode to be used during the CSAC experiment.

5 CSAC Breakout Board

Figure 9: Assembled Breakout Board

The past semester focused on the development of a functionally similar, low-cost alternative to the Microsemi SA.45s Chip Scale Atomic Clock (CSAC) Developer's Kit. The primary purpose of the design was to create a compact and easy to use alternative for the CSAC Developer's Kit, which is somewhat cost prohibitive and does not meet all of the requirements of the CONTACT project. The following subsections elaborate the steps involved in the breakout board design and testing.

5.1 Defining Requirements

The design of the breakout board is primarily based on the reference schematic provided by Microsemi for their Developer's kit. All the core digital logic circuitry, the power supply and

filtering circuitry, and general layout guidelines were based on Microsemi's board design, and were copied over without any major modifications.

In addition to the core logic and power circuitry copied over from Microsemi, some additional features were added to the design to facilitate CSAC testing. These additional features included a mini USB connector and a USB FTDI converter to power and enable communication with the CSAC over the same cable, an additional 10 MHz output derived from the CSAC output, and some ESD protection circuitry in the form of an ESD protection diode to protect the CSAC.

5.2 Hardware Design

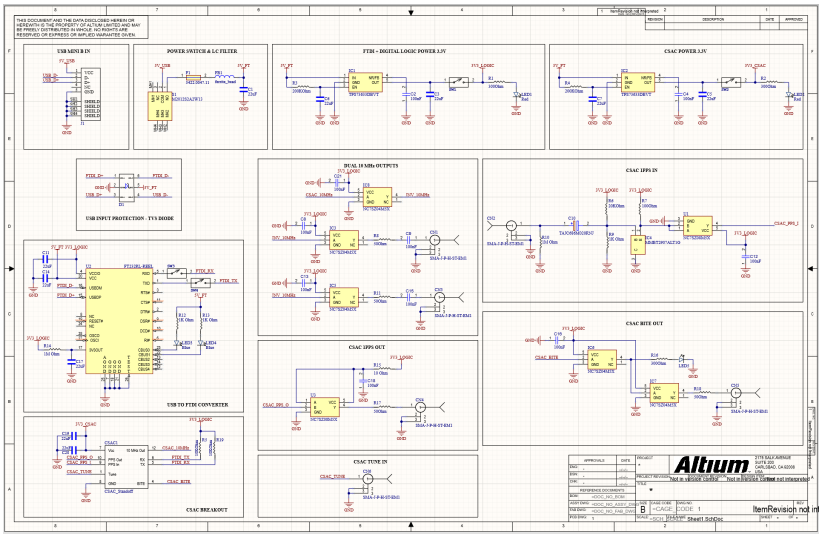


Figure 10: Breakout Board Schematic

Altium designer was used to develop the schematic of the breakout board. The design shown in Figure 10 is the final schematic of the breakout created after combining essential parts of Micosemi's reference design and the additional features added to the board.

To power the board, there are two low dropout 3.3 volt linear voltage regulators powered by +5V provided by the USB connector, which is then filtered with a ferrite bead. One regulator is used to power the CSAC, and the other is used to power all other digital logic. The FT232 USB to UART converter is used to communicate with the CSAC over USB. This eliminates the need for a separate +5V adapter to power the board and a seperate RS232 cable connected to a USB to RS232 converter to enable communication with the CSAC. The TVS diode connected to the data lines of the USB connector should protect the board against accidental ESD discharges.

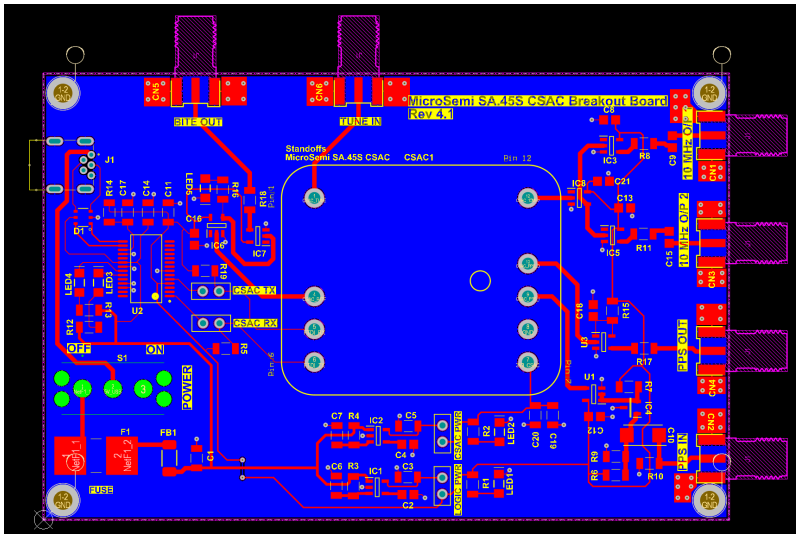


Figure 11: Breakout Board Layout

The layout, shown in Figure 11, was created using Altium Designer as well. To ensure low noise the USB inputs and the communication chip were placed away from the CSAC and the LDOs. Multiple decoupling capacitors were placed to filter out some of the noise, and the 10 MHz output traces were designed to have 50Ω traces and source series terminations to prevent reflection noise. The final layout was about 37% smaller than the Microsemi Developer's kit.

5.3 Testing the Breakout Board

To prevent damaging the CSAC while initial power-up, testing the breakout board was done in stages.

First, after the assembly was completed, a simple continuity test between the pins of the CSAC and the power pads was done to help detect any shorts in the board. Once that was done, and no direct shorts were detected, the board was powered on without the CSAC placed on the board. After powering the board for the first time, voltages were checked at the power pins of the CSAC to ensure that they are within operating specifications.

To simulate CSAC operation, a function generator was used to generate a reference 10 MHz and 1 PPS signals. These signals were fed into the digital logic used to condition the CSAC outputs, and the resultant waveform was tested with an oscilloscope. Once testing the digital logic was completed, the TX and RX pins of the USB-FTDI converter were shorted to echo the input, hence testing the USB to serial converter. After all the aforementioned tests were completed, the CSAC was inserted into the socket and all the tests were repeated.

The CSAC breakout board shows up as a COM port when plugged into a USB port,

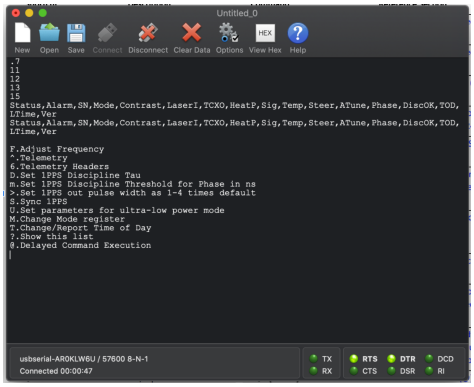
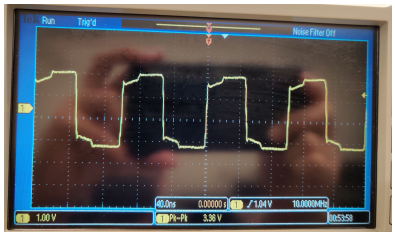
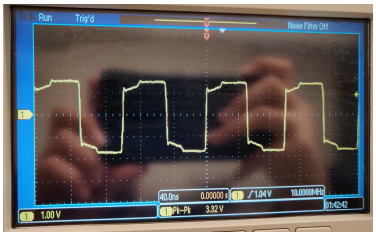


Figure 12: CSAC/Breakout Board Serial Port Communication

and a terminal emulator (e.g. puTTY) can be used to communicate with the board. On inserting the CSAC and connecting to it, a help menu shows up when the key 'H' is pressed, which is shown in Figure 12. As the CSAC responds to a key press, we can confirm that bidirectional communication with the CSAC is possible.



(a) Breakout Board 10 MHz output



(b) Microsemi Board 10 MHz output

Figure 13: 10 MHz Output Comparison

On comparing the 10 MHz output from the breakout board (Figure 13a) and the Microsemi Developer's kit Figure (13b), we notice that they are identical. The breakout board has a slightly higher peak-to-peak voltage (40mV), but that might just be an effect of using a different power source to power the board. This slightly higher noise should not affect operation, as the frequency readout from the oscilloscope was a stable 10 MHz, and there is no noticeable noise or distortion around the zero crossings.

As the new breakout board is compatible with the CSAC, and the input and outputs are identical to the Microsemi Developer's kit, the breakout board is ready to be used with the testbed.

6 Filter and Clock Ensemble Subsystem

The development of the filter and clock ensemble subsystem this semester has been focused on processing real phase measurements and validating filter performance. This section will be broken down into hardware configuration, data acquisition, and filter performance subsections and will include some hardware configuration work performed during the summer months which has not been reported previously.

6.1 Hardware Configuration

In the Spring 2020 semester the clock ensemble was implemented in GNU and tested using only simulated data. The focus of the further development of this subsystem during the summer months was to achieve CSAC phase measurements on the SDR for the purpose of validating filter performance.

A series of tests were conducted to characterize the Ettus TX/RX and RX channels and to understand sending commands to and acquiring data from the Ettus through GNU Radio. A sample of one of these tests is shown in fig 14. In this test a 10 MHz signal is generated from a DSOX1204G Oscilloscope Gen Out channel. The signal is connected to the Ettus RX2 port and sampled at 25 MS/s. The UHD USRP source block is set to a 10 MHz center frequency and the GUI Sink is set to a center frequency of 0 Hz, so it is expected to see a spike at -10 MHz and 0 Hz.

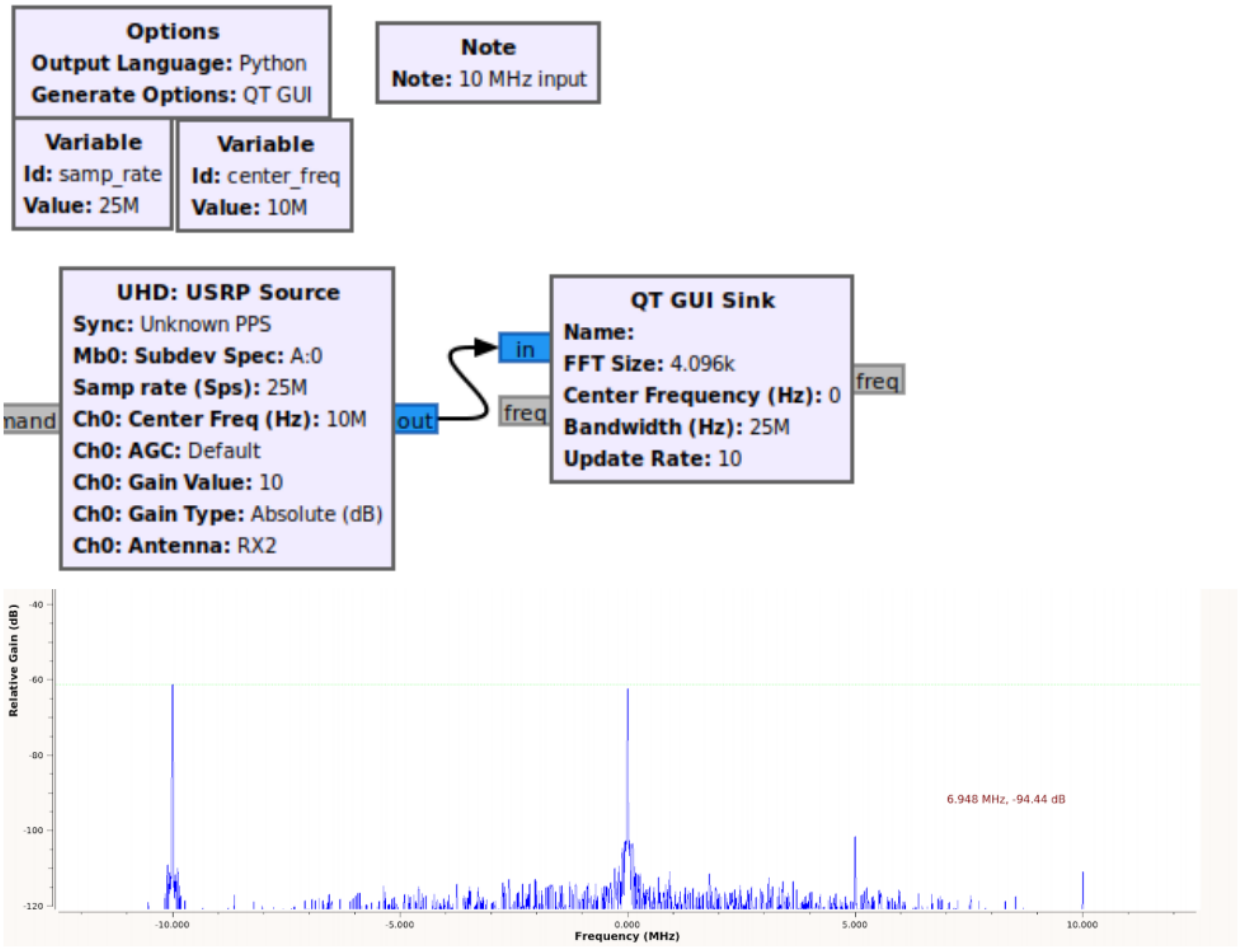


Figure 14: Ettus RX Trial

After conducting tests to gain some insight into controlling the Ettus, the Microsemi CSAC dev board was connected to the Ettus with 30 dB attenuation to generate 26 mVrms on a 50 Ohm load. The center frequency is set to 10 MHz - 100 Hz, with a sampling rate of 250 kS/s and a decimation of 128. The signal peaks at 100 Hz as expected and is displayed in the GNU GUI as shown in fig 15.

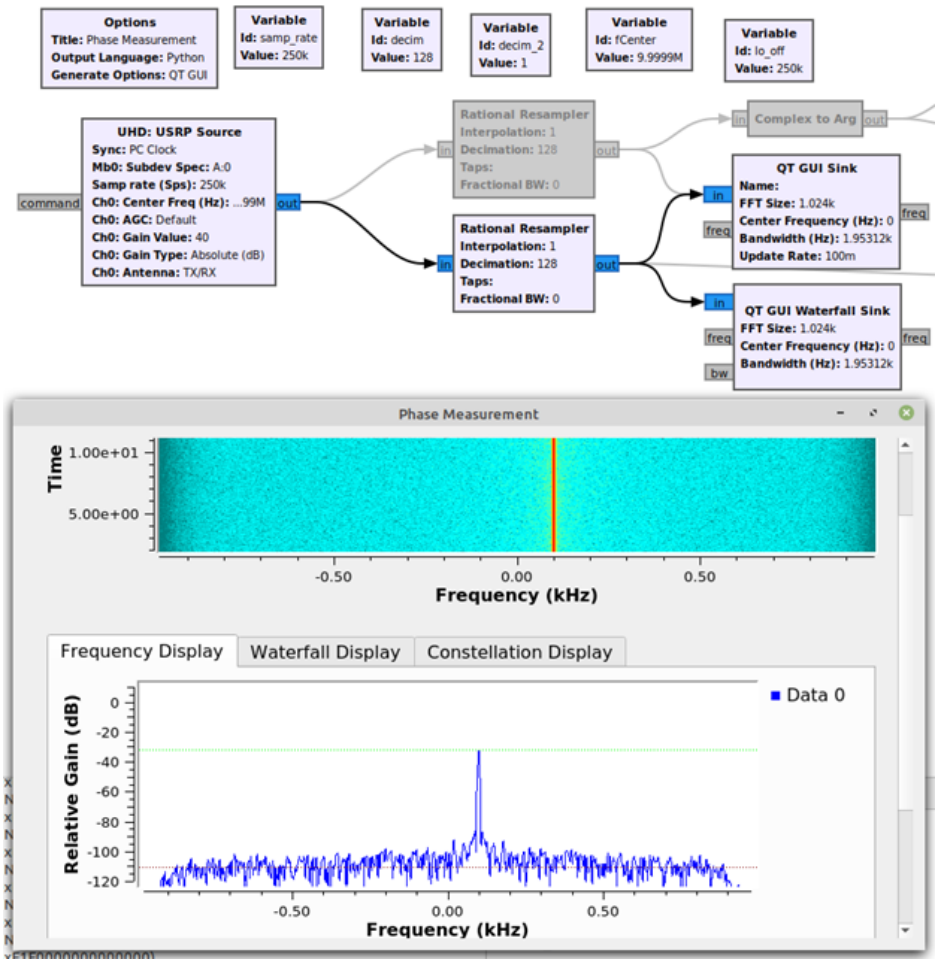


Figure 15: CSAC Measurement in GNU Radio

6.2 Data Acquisition and Filter Performance

Several data sets were used to test and validate filter performance. A long-term CSAC measurement test had not been run on the Ettus due to file size ease-of-use, so to evaluate filter performance a CSAC data set taken from the NovAtel 729 receiver in Fall 2019 was selected. Two 24-hr CSAC runs were available, so to generate a data set consistent with testbed performance expectations, the phase data was split up to generate 3 CSAC runs of approximately 11 hours. The phase measurements of CSAC 2 and 3 were differenced with CSAC 1 to generate the phases shown in fig 16.

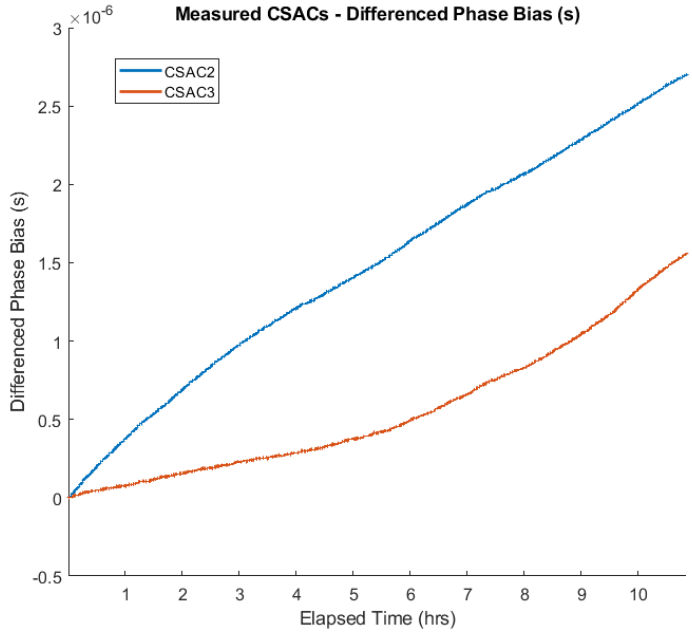


Figure 16: Differenced CSAC Phases

The differenced phase measurements for CSAC 2 and CSAC 3 are the inputs into the Kalman filter, and the outputs are phase and frequency estimates for all three clocks. The phase estimate outputs are consistent with expected statistical behavior and are shown in figure 17.

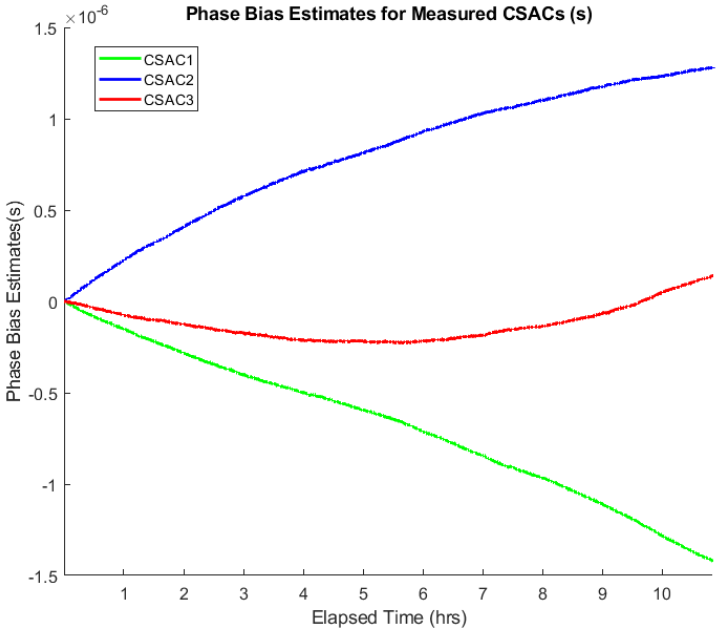


Figure 17: CSAC Phase Estimates

Looking ahead, the next steps for the development of the filtering subsystem is to operate in real-time for a 24-hour CSAC test on the Ettus. Additionally, it should be integrated and tested with the error injection subsystem.

7 Signal Generation Subsystem

Development for the signal generation subsystem over the past semester was focused on analyzing the feasibility of implementing the subsystem design with the current software defined radio being used to host the testbed. Given the architecture of the N310 USRP, signal generation operations would have to be performed at baseband or a "near-baseband" intermediate frequency (IF) - between the front-end upconversion and downconversion chains. While high performance frequency steering has been demonstrated for antenna beam forming applications by [2], we have not found evidence of this approach applied to the steering of timing signals. The primary concern with this idea of steering at or near baseband is the lack of clock synchronization between the up/downconversion chains with the N310 daughterboards. If timing is not synchronized, this could add phase error to the output signal. However, if these clocks are synchronized and perfect up and downconversion is assumed, is there any difference in steering performance? To answer this question and gain a deeper understanding of the fundamental differences between the two steering approaches, simplified experiments were conducted with low frequency signals and a *National Instruments* myRIO digital controller. Additionally, these experiments were used as a means to demonstrate functionality of the nominal and baseband steering system designs and the ability to implement them with hardware.

7.1 myRIO Steering Experiments: Setup

The myRIO is a digital controller on which Real-Time (RT) systems can be built and deployed using LabVIEW as a development environment. Real-Time steering algorithms were developed for the nominal ("high-rate") and N310 ("baseband") steering approaches. The algorithms for both systems were tuned to steer an externally generated 53 Hz sine wave to 54 Hz over a 200 second time interval and maintain the 1 Hz difference for an additional 2800 seconds. Data was collected at a rate of 400 S/s with an Arduino Uno microcontroller and the external input signal was provided by a Koolertron CJDS66 function generator. A high level block diagram for the the structure of the high-rate system implementation is shown in Figure 18.

Observable in Figure 18, the high-rate system is essentially a direct low frequency im-

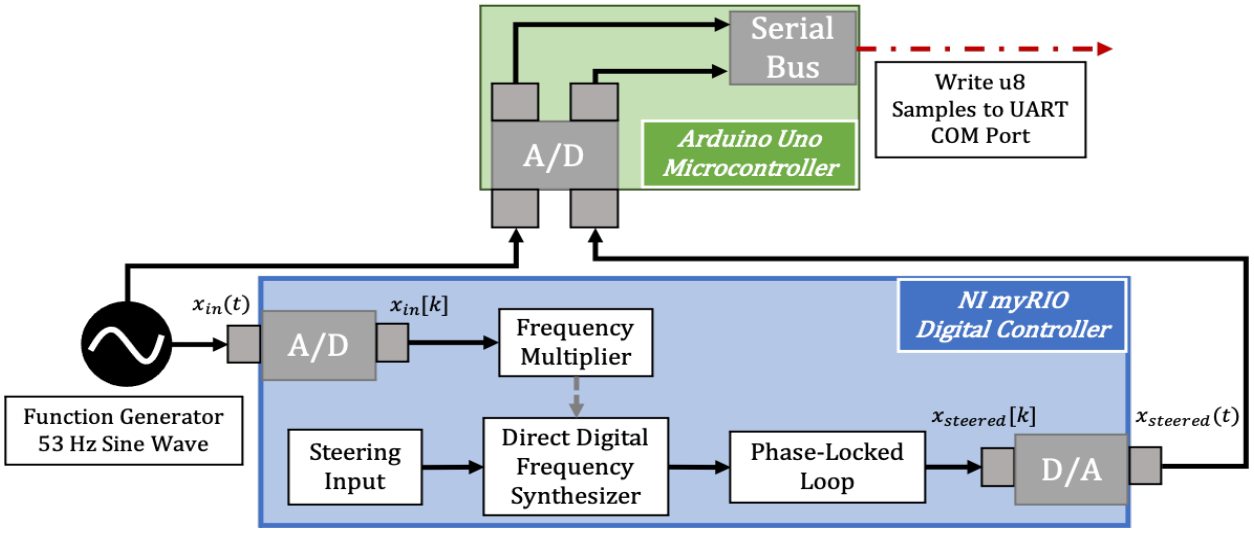


Figure 18: Block diagram for the high-rate steering system implementation.

plementation of the nominal subsystem design, except the steering input estimator and regulator were replaced with a ramp function. The baseband implementation is similar to the high-rate system, but has the additional synchronized downconversion and upconversion chains. The high level block diagram for the structure of the baseband implementation is provided in Figure 19.

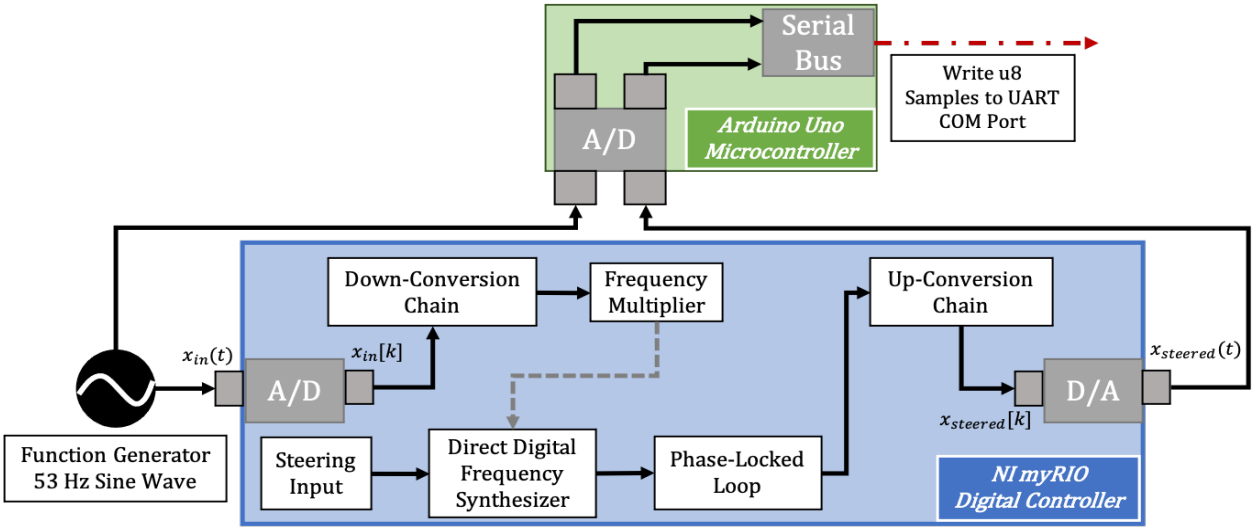


Figure 19: Block diagram for the baseband steering system implementation.

The downconversion chain in the block diagram above consists of sine/cosine mixers and low pass filters set to shift the 53 Hz signal down to a 5 Hz complex signal. The upconversion chain in Figure 19 uses the complex transmitter method shown in [3], where the Q component of the baseband steered signal is shifted by 90^o, prior to sine/cosine mixing, to produce a

single sideband (SSB) upconverted steered output. The steering input parameters for each implemented system under test are shown in Table 4.

Method	Δf Step Size	Relative $\frac{\Delta f}{f_n}$	Steering Interval
High-Rate	0.005 (Hz)	$9.4x10^{-5}$	1 step/s
Baseband	0.005 (Hz)	$1.0x10^{-3}$	1 step/s

Table 4: Steering parameters for each system demonstration with the myRIO controller. The frequency adjustment size is the same for both systems, but the baseband method introduces a larger relative fractional frequency adjustment size.

The hardware configuration for the myRIO is as follows - based on manufacturer specifications [8]. The 12 bit A/D converter runs at 500 kS/s and continuously samples the analog input voltage. These samples are read-in by the Xilinx Zynq-7010 FPGA. The RT steering algorithms run within 1 kHz timed loop structures and are deployed onto the FPGA ARM processor. Each loop iteration, a sample is retrieved from the FPGA, operated on, and passed back to the FPGA - where they are written out to the 12 bit D/A running at 345 kS/s. A block diagram of the hardware configuration for the experimental setup is displayed in Figure 20.

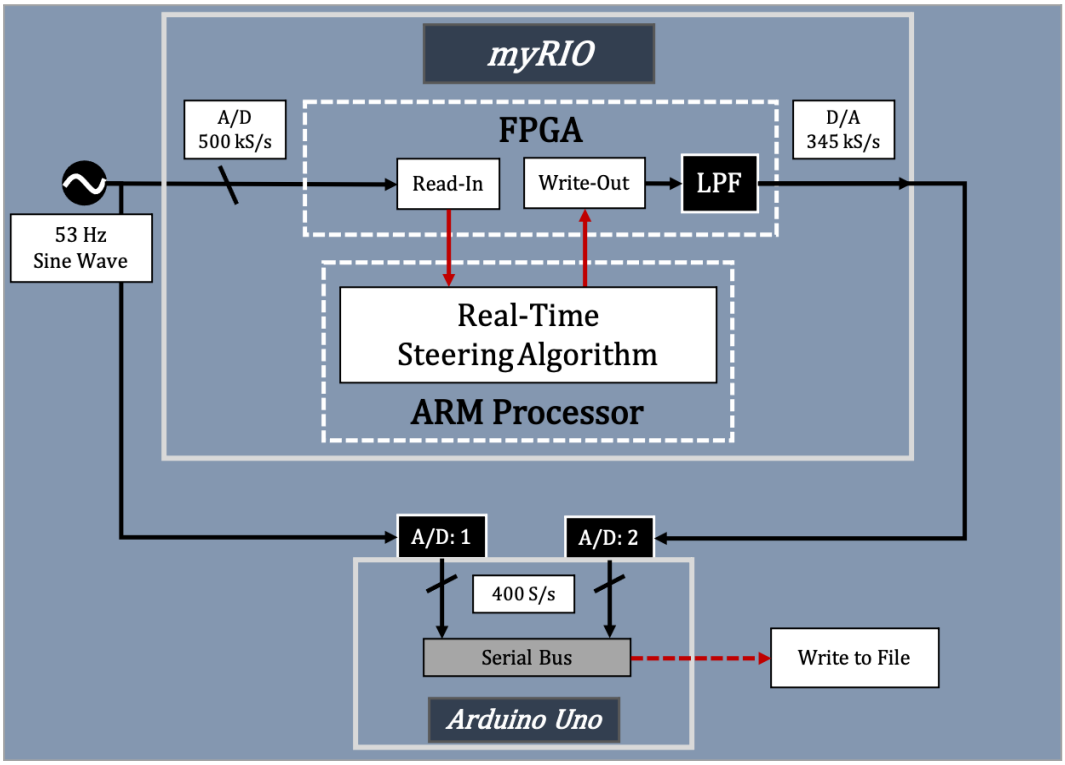


Figure 20: myRIO steering experiments hardware configuration block diagram.

In order to improve out of the box performance, LabVIEW FPGA and Xilinx cloud compilation were used to customize the FPGA and boost processing speed, as well as low pass filter the steered samples before writing to the D/A. The following subsection discusses results from the low frequency signal steering experiments.

7.2 myRIO Steering Experiments: Results

Data sets collected by the Arduino Uno were post-processed in MATLAB. The comparison between the two steering methods tested was broken up into two areas of analysis: short-term performance and long-term performance. For short-term performance comparison, spectral analysis and beat signal quality were used as the performance metrics. For the long-term performance comparison, phase measurements were computed using methods described by [9], and Overlapping Allan Deviations (OADEV) were used as the performance metric.

Beginning with the short-term performance comparison, spectral analysis was used to inspect power across the frequency spectra of the input and steered output signals, to confirm steering accuracy and assess sideband noise. Power spectral density (PSD) was computed by taking Fourier Transforms of the signals, after steering had completed and over a span of 250,000 sample points. This sample point span gives the results a resolution bandwidth

(RBW) of 0.0016 Hz and is calculated as:

$$RBW = \frac{f_s}{n} = \frac{400\ Hz}{250000} = 0.0016Hz \tag{1}$$

The spectral analysis results for the high-rate steering system are provided below in Figure 21.

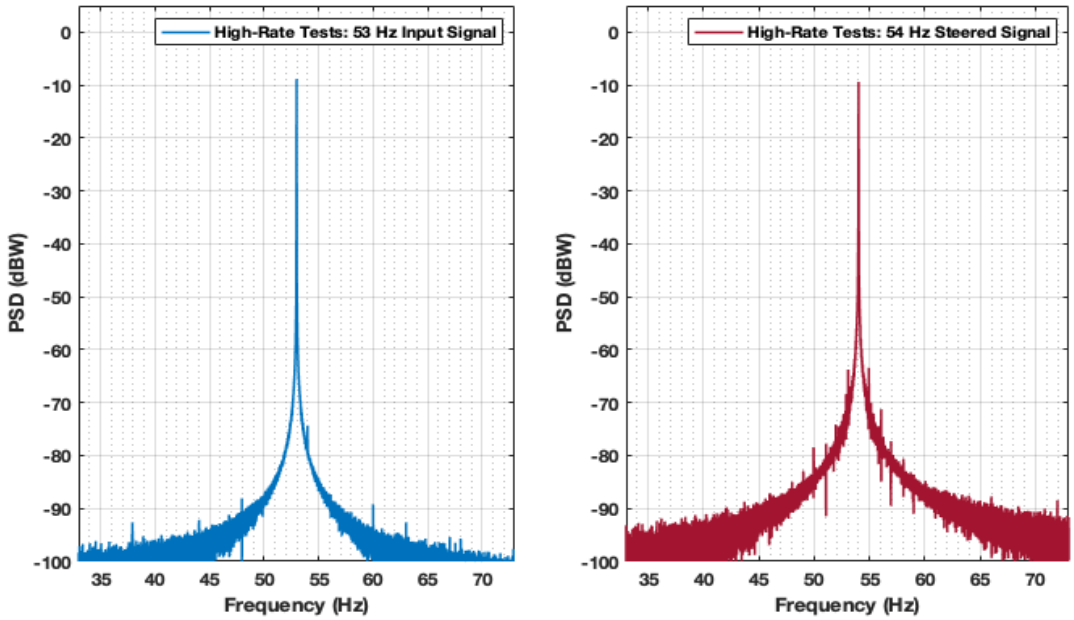


Figure 21: Spectral analysis results for the high-rate steering system.

PSD results from the high-rate steering experiments are as desired. The input and steered output signals both have very clean frequency spectra, and power is focused exactly at 53 Hz for the input and 54 Hz for the steered output. As well, sideband noise in the steered output is below 60 dBW. The baseband steering system results also show satisfactory short-term frequency performance. Again, power in the input signal and steered output signal is focused exactly at 53 Hz and 54 Hz, respectively. While there is slightly more sideband noise present in the steered output, power in the sideband noise is still below 60 dBW. Spectral results for the baseband steering system are shown in Figure 22.

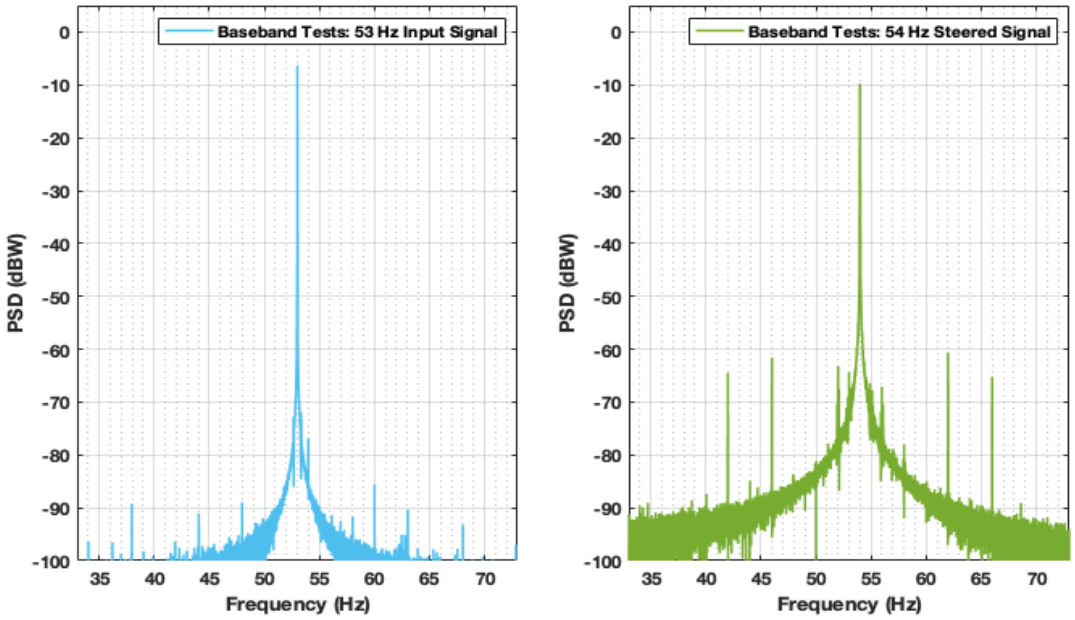


Figure 22: Spectral analysis results for the baseband steering system.

For the first part of the short-term performance comparison, spectral analysis results show that the two steering methods are fairly comparable. The secondary part of the short-term performance assessment is the comparison of beat signal quality. The beat signals were formed by mixing (or "beating") the input signals with/against the steered output signals and low pass filtering to remove the high frequency term produced by the mixing operation - shown in Equation 2.

$$2sin(2\pi(53\ Hz)t_k)sin(2\pi(54\ Hz)t_k) = cos(2\pi(1\ Hz)t_k) - cos(2\pi(107\ Hz)t_k) \tag{2}$$

The beat signals were formed in MATLAB through element by element vector multiplication and filtered with a discrete-time 2nd order butterworth LPF, with a cutoff frequency of 10 Hz. The discrete-time transfer function used to define the beat signal filter coefficients is given by:

$$H(z) = \frac{0.005846z^{-1} + 0.005548z^{-2}}{1 - 1.832z^{-1} + 0.8546z^{-2}} \tag{3}$$

Assessment of beat signal quality provides a means to compare steering quality between the two methods, as well as a secondary check on steering accuracy. Given the steering input parameters, desired time-series results should show a smooth evolution of the beat signals into a clean 1 Hz sine wave - after the steering interval has completed. The time-series results comparison of the beat signals from each steering method, over the steering interval,

is shown in Figure 23.

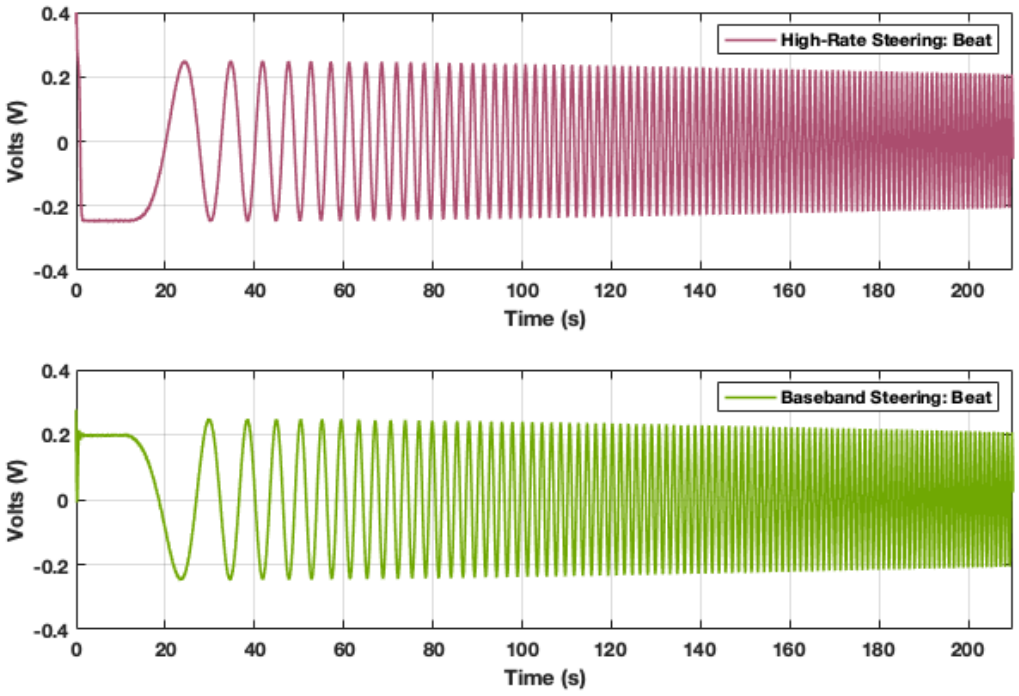


Figure 23: Beat signal comparison over the 200 second steering interval.

Figure 24 shows a zoomed in view of the beat signals after steering has completed.

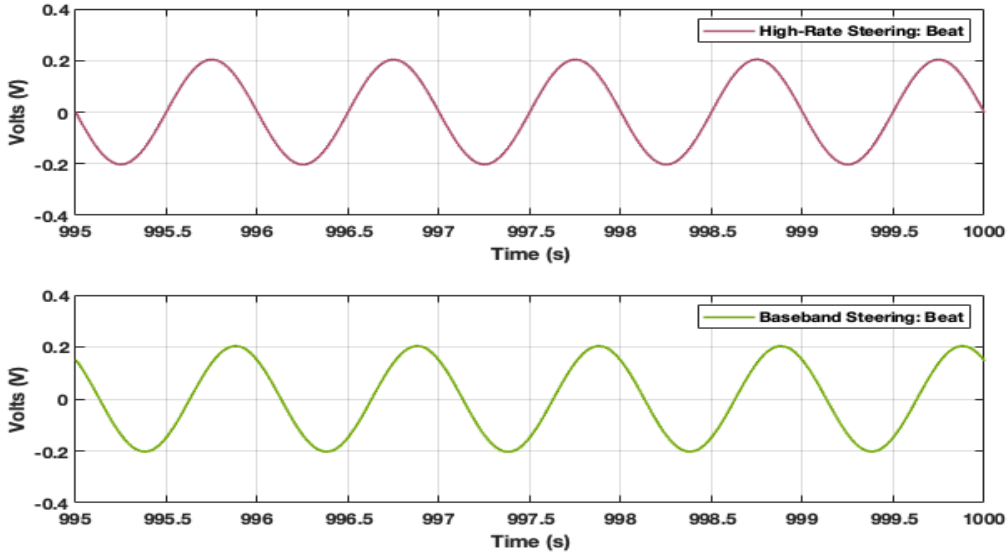


Figure 24: Beat signal comparison after steering has completed.

The beat signal results over the steering interval show desired and comparable performance for both steering methods, with smooth transitions into a 1 Hz sine wave. As well, the time-series results, once steering has completed, also show clean sine waves with a period of exactly one second. These results confirm what was observed from spectral analysis and indicate that there is no significant difference in short-term performance between steering methods.

As mentioned earlier, long-term performance was the second portion of the results analysis. This analysis quantified frequency stability over long time intervals. OADEV was the analysis tool used to compare frequency stability of each steered output signal, with respect to the input signal that was steered. OADEV's were computed from the normalized phase-time deviations calculated for each signal, after removing the effect of the Arduino sample clock. Results shown in Figure 25 display the OADEV results from two different tests for each implemented steering system.

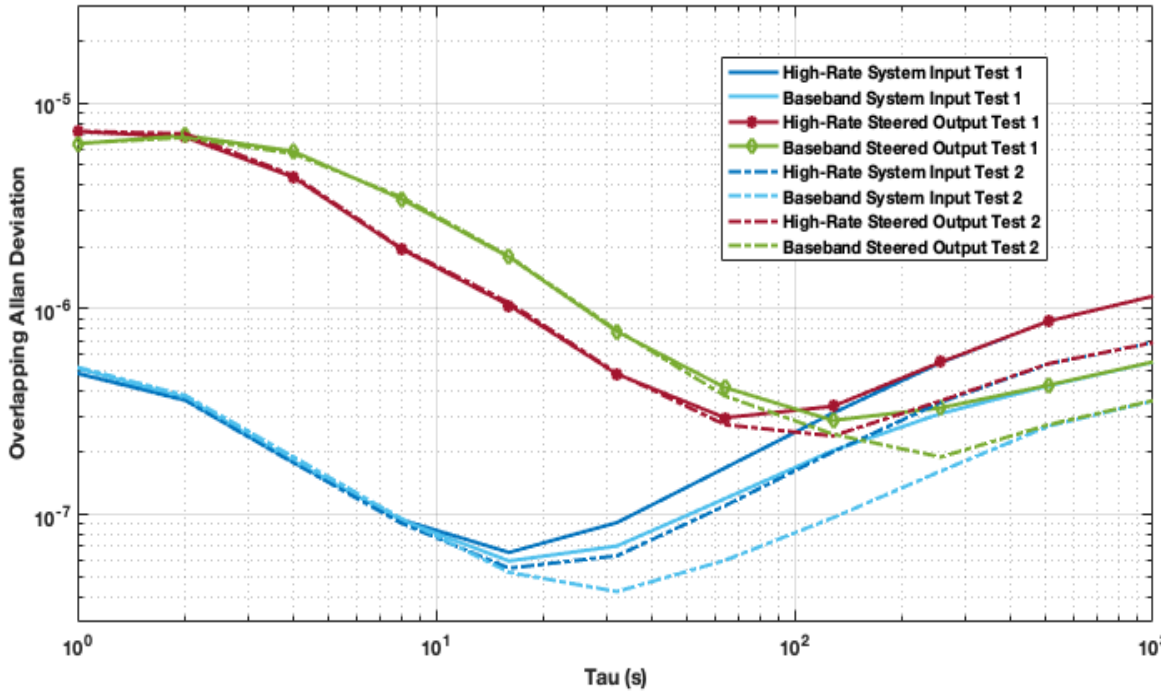


Figure 25: Overlapping Allan Deviation result comparison between steering methods.

The long-term stability results show that initial values match up fairly well with the specified stability of the function generator clock ($\pm 1\,ppm$) and the Z-7010 FPGA clock ($\pm 10\,ppm$). As well, at longer time intervals the steered outputs track their respective input signals. A slight difference in frequency stability is observable between the two methods. Steered outputs from the high-rate system appear to perform better than the baseband

system outputs by approximately a factor of 2. While there is a difference present in these results, baseband steering does not appear to significantly degrade short-term or long-term steering performance.

8 Measurement Subsystem

The measurement subsystem makes phase measurements on the clock signals and enables the characterization of the individual clocks. The signal characteristic that contains information about the oscillator stability is the signal phase. The phase is computed by taking the complex argument of the I & Q components of the clock signals. The difference between this measured phase growth and the phase growth of an ideal signal shows the frequency offset of the oscillator. A block diagram showing the measurement system process is shown below.

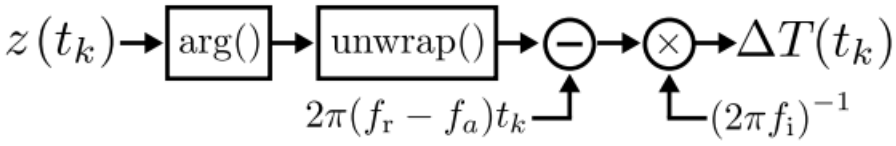


Figure 26: Measurement System Block Diagram [9]

In figure 26 above, $z(t_k)$ represents the complex signal data that is passed into the argument function. That wrapped phase is then unwrapped, the ideal linear phase growth is removed, and the remaining data is used to calculate the time offset based on the measured phase difference. The measurement system implements the above logic via custom python blocks in GNU Radio Companion.

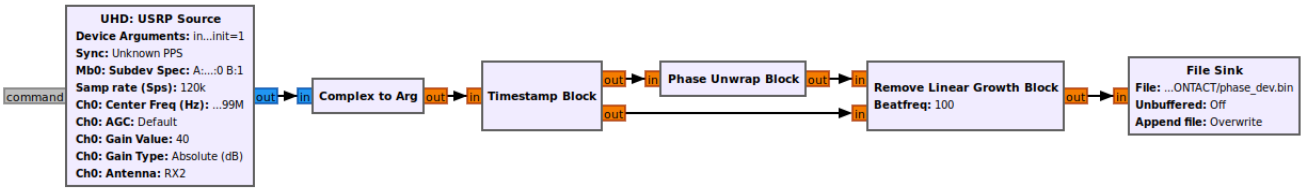


Figure 27: Measurement System Block Diagram Implemented in GNU Radio Companion

8.1 Measuring Phase Growth

A number of data collection experiments were performed to make phase measurements of input signals. One such test used identical signals from a Rubidium frequency standard to characterize the difference between the various RX processing chains on the software defined

radio. The Rubidium signals were connected to RX ports on the same daughterboard and the sampled data went through the signal processing pipeline in figure 27.

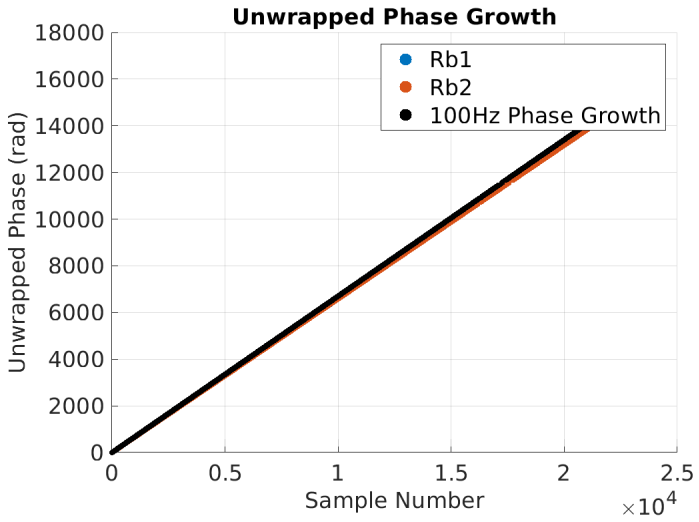


Figure 28: Unwrapped Phase from Identical Rubidium Input

Figure 28 above shows the measured phase growth of the two Rubidium input signals and the values for the ideal phase growth of a 100Hz signal. The phase values for each of the Rubidium signals are indistinguishable at this level since the input signal is the same for both RX channels. The dominant contributor to the phase growth in this image is a result of the signals having a frequency of ≈100Hz. The 100Hz signals are due to the desired beat frequency that the SDR tunes the input 10MHz signals to. Analyzing these lower frequency signals vastly reduces the storage requirements for long term signal characterization while simultaneously preserving the signal behavior of interest. The next step in the processing chain, removing the ideal 100Hz phase growth, should eliminate the 100Hz contribution to the phase growth and result in phase deviations more reflective of the oscillator stability.

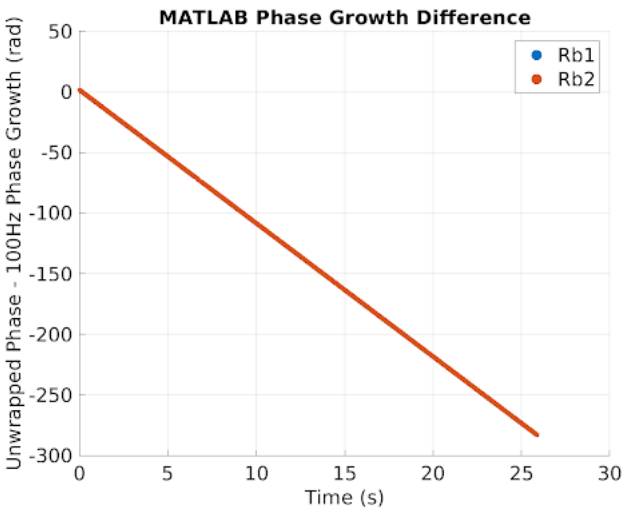


Figure 29: Unwrapped Phase with Ideal Phase Growth Removed

Figure 29 above shows the measured phase growth of the Rubidium oscillators with the 100Hz phase growth values removed. Both Rubidium signals are still indistinguishable at this level, which is a good sign since the signals are identical. Upon further inspection, the phase deviation described by this graph is much too large given the stability specifications of Rubidium oscillators. A phase deviation of -275rad over the course of 26 seconds yields an angular frequency offset of: $\frac{\Delta\phi}{\Delta t} = \frac{-275rad}{26s} = \omega = 10.58\frac{rad}{s}$. Converting to frequency offset yields $f = \frac{\omega}{2\pi} \approx 1.7$Hz. The corresponding fractional frequency for this frequency offset is $\frac{\Delta f}{f_{nom}} = \frac{1.7}{10*10^6} \approx 1.7 * 10^{-7}$. This fractional frequency offset is much too large for the Rubidium frequency standard, so something else must be concealing the behavior of the Rubidium signal.

There are a number of different oscillators on the SDR that are used for digital and analog signal processing. The dominant contributor to the phase growth in figure 29 is due to these oscillators, not the Rubidium reference. One method that should eliminate the contributions of a common clock is to difference the phase measurements.

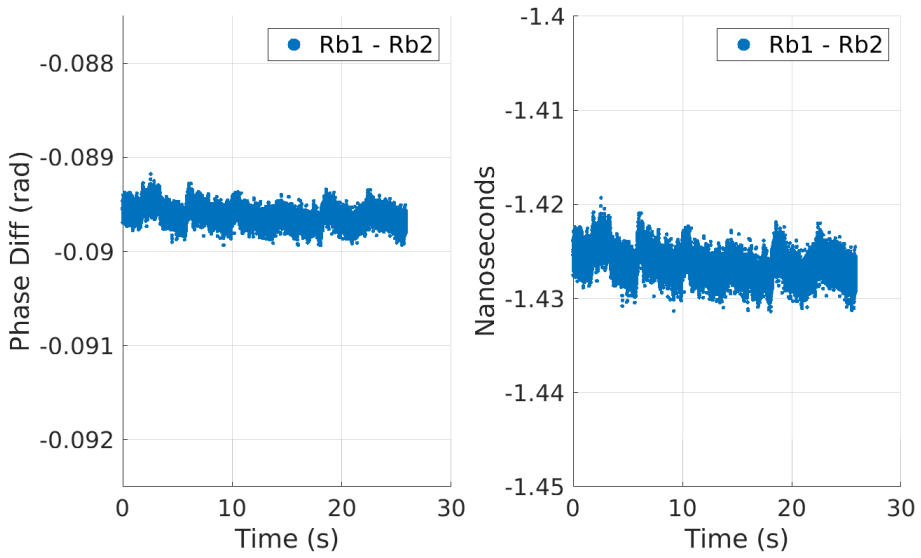


Figure 30: Difference Between Rubidium Signals: Same Daughterboard

The figure above shows the result of differencing the two Rubidium signals in figure 29. The phase deviations of this signal are much less pronounced than before, indicating that the common clock error has been removed along with any other common error sources on the same daughterboard.

8.2 Daughterboard Noise Comparison

The previous section showed that error contributions common to signals on the same daughterboard cancel out. However, to accomplish the project goal of measuring and ensembling three or more clocks, both daughterboards of the SDR need to be used. Phase measurements made on different daughterboards will ultimately be sent to a Kalman Filter and it is important to understand the measurement noise associated with each measurement. A study similar to the one described in the previous section was performed to assess the noise characteristics of signals on the same daughterboard and signals on different daughterboards.

In this experiment a split Rubidium signal was used as two identical inputs to the SDR RX chains. The data gathered from the SDR regarding the input signals was the raw, low frequency signal information so as not to conflate errors in the GNU Radio signal processing with error contributions from the SDR. The six total combinations of signal inputs are shown below. The left most column corresponds to the signals that were differenced on the same daughterboard and are three orders of magnitude smaller than the cross-board residuals. The center and right most columns describe the signal residuals for signals that were differenced on different daughterboards. All of the histograms below had the mean subtracted for ease of comparison.

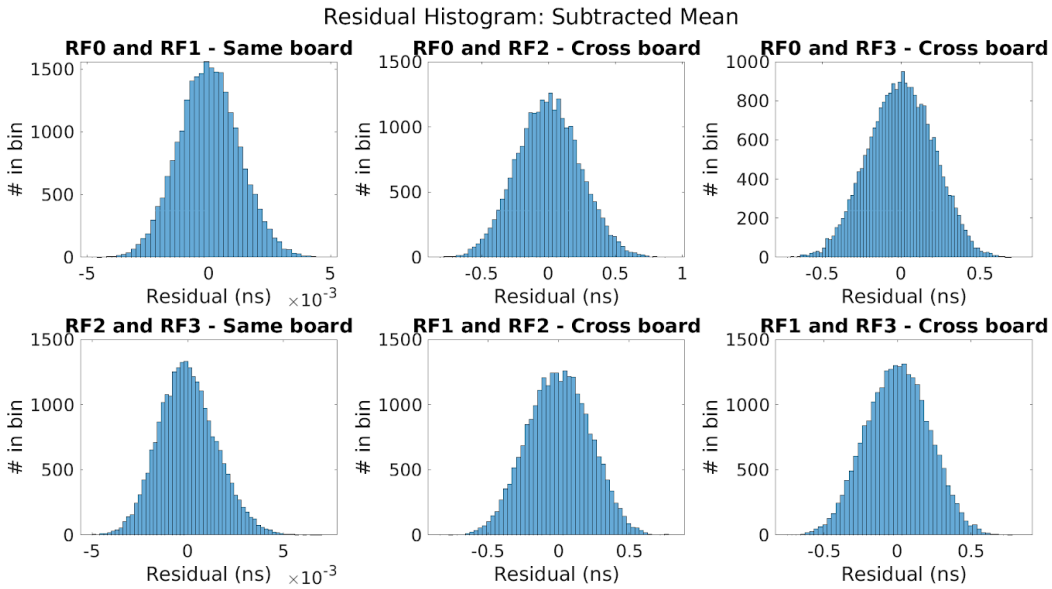


Figure 31: Residuals for Signal Differences Across SDR Inputs

8.3 Measurement System Summary

The Ettus N310 is capable of making phase measurements of the 10MHz oscillators. However, these standalone measurements contain many different sources of error from the SDR signal processing chains that affect the quality of the phase measurements. Some of these errors are common across daughterboards and on the same daughterboard. When the phase measurements are differenced, the resulting data contains phase information that is much more reflective of the stability of the oscillators used in this project. This result demonstrates the potential viability of the Ettus N310 as clock ensembling testbed.

9 Error Injection and Detection

9.1 Error Detection

The error detection system is required for the clock ensemble testbed to identify clock errors in the measured CSAC signals in real-time. Errors in the clocks will directly impact the state estimates output by the filtering and ensembling system. The erroneous clock state estimates are provided to the signal generation system. Ultimately, these errors will appear in the steered output signal. Detecting these errors is an essential first step toward mitigate their effects on the steered output signal. Separate fault management algorithms will be developed to enhance the robustness of the generated signal.

A Kalman filter (KF) jump detection algorithm was implemented in GNU radio during a previous semester. This custom GNU radio block compares measured fractional frequencies to fractional frequency estimates in order to detect common clock errors (i.e. frequency and phase jumps). With that said, the custom Python code contained a few errors. Moreover, the Kalman filter was not tuned to properly to accurately model the Ettus N310 front end. A brief outline of the KF jump detection algorithm is given below. The algorithm is adapted from the KF system outlined by Gödel [6]. For the following section, x refers to a fractional frequency value.

The KF registers a jump in a fairly straightforward manner. During each iteration of the KF, an estimated fractional frequency, $\hat{\mathbf{x}}^-(t_k)$, is calculated based on the previous fractional frequency estimate, $\hat{\mathbf{x}}(t_{k-1})$. For a stable timing signal, the fractional frequency should ideally remain constant over time (Eq. 4). Therefore, the state transition matrix defined for the KF is simply a scalar value of one (Eq. 5). In Eq. 5, τ is the measurement interval in seconds.

$$\hat{\mathbf{x}}^- = \Phi(\tau)\hat{\mathbf{x}}(t_{k-1}) \tag{4}$$

$$\Phi(\tau) = 1 \tag{5}$$

However, in reality, we expect the value to differ from the expected fractional frequency due to both clock errors and measurement noise. Since only one state is being analyzed, the error covariance, P, is a scalar-valued function of the process noise variance, $Q(\tau)$. The initial error covariance before the correction step, $P^-(t_k)$, is given by Eq. 6. Note that $P(t_{k-1})$ is the error covariance after the correction step at time t_{k-1}. The process noise, $Q(\tau)$, was determined to be 1×10^{-20} based on the CSAC OADEV analysis performed in previous semesters. This value is taken from the slope of the OADEV curve for averaging windows, τ, of one second.

$$P^-(t_k) = \Phi(\tau)P(t_{k-1})\Phi(\tau)^T + Q(\tau) \tag{6}$$

The Kalman gain, $K(t_k)$, is defined based on the error covariance, $P^-(t_k)$, and measurement noise variance, $R(t_k)$ (Eq. 7). In the absence of simulated clock errors, the fractional frequency data fluctuates with an amplitude of around 5×10^{-10}. In its current implementation, the process noise covariance must account for the majority of the noise in the measured clock signal.

$$K(t_k) = P^-(t_k)[P^-(t_k) + R(t_k)]^{-1} \tag{7}$$

The estimated fractional frequency, $\hat{\mathbf{x}}^-$, is corrected using this Kalman gain and the difference between the measured ($z(t_k)$) and initial estimated fractional frequencies (Eq. 8). The corrected error covariance, $P(t_k)$, is given by Eq. 9. Again, only one state is considered, so $P(t_k)$ is a scalar.

$$\hat{\mathbf{x}}(t_k) = \hat{\mathbf{x}}^-(t_k) + K(t_k)[z(t_k) - \hat{\mathbf{x}}^-(t_k)] \tag{8}$$

$$P(t_k) = [1 - K]P^-(t_k) \tag{9}$$

Now, the Kalman filter is set up to predict the evolution of the fractional frequency data. A simple residual detector is used to detect jumps in the fractional frequency data. The residual, $r(t_k)$, is defined as the difference between the measured fractional frequency, $z(t_k)$, and the predicted fractional frequency prior to the correction step, $\hat{\mathbf{x}}^-(t_k)$ (Eq. 10). The residual actually appears in the correction step in Eq. 8.

$$r(t_k) = z(t_k) - \hat{\mathbf{x}}^-(t_k) \tag{10}$$

The KF assumes that the residuals are normally distributed about a zero mean with a variance equivalent to the sum of the process and measurement noises. Thus, the standard deviation of the residuals is given by Eq. 11.

$$\sigma_{r(t_k)} = \sqrt{P^-(t_k) + R(t_k)} \tag{11}$$

At this point, we have a residual which compares the measured fractional frequency to the estimated fractional frequency at a given sample epoch. The modeled standard deviation of the residuals describes the range of probable values. To be specific, one would expect greater than 99.99 percent of residuals to fall within four standard deviations of the zero mean. Residuals outside of four standard deviations are statistically significant. Residuals this large likely correspond to clock errors present in the signal. The measured phase deviation during a sample interval will be much larger in the presence of clock errors. Thus, when a frequency or phase jump occurs, the fractional frequency computed over the current averaging window will be offset from the previous fractional frequency. If this change in the measured fractional frequency results in a residual greater than four standard deviations from the mean, then a 'jump' is flagged. Therefore, this value of four standard deviations of the residuals sets what is referred to as the 'jump threshold'.

This simple Z-test is capable of effectively identifying jumps in measured clock signals. However, a measurement noise variance, which accurately describes the system, must first be

selected. The measurement noise can be approximated through inspection of the fractional frequency data. For this analysis, two measured clock signals are utilized: one downconverted 100Hz CSAC signal and one downconverted 100Hz Rb signal. Each signal is sampled at a rate of 1953.125Hz. The fractional frequency of the CSAC signal is computed over approximately one second long averaging windows, using the Rb signal as a reference clock. The computed fractional frequencies are depicted in figure 32. The Allan deviation for a one second time intervals (9×10^{-11}) is included as well. The ADEV for $\tau = 1$s is slightly better than the manufacturer reported value of 3×10^{-10}.

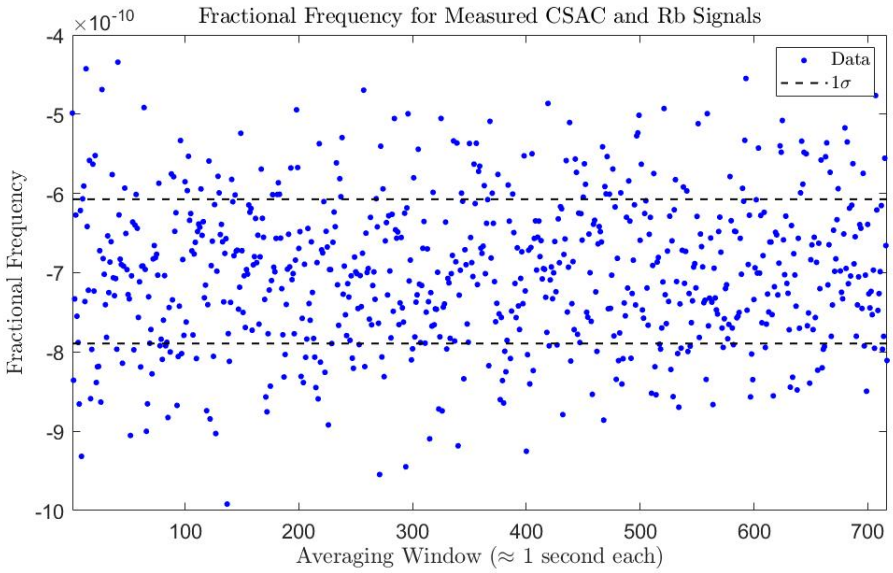


Figure 32: Measured CSAC Fractional Frequency Data

The variance in the fractional frequency captures both the process noise and the measurement noise present in the system. The computed Allan deviation indicates that the measurement noise variance is likely on the order of 10^{-21}. An iterative approach to solving for a measurement noise variance is employed.

First, an initial guess for the value for the measurement noise variance is chosen. Then, the residuals of the KF jump detection block are recorded for further analysis. The modeled standard deviation of the residuals (Eq. 11) is compared to the experimental standard deviation of the residuals. If the two standard deviations are offset, then the estimated measurement noise variance is not a good fit for the model. The initial guess is updated and the process is repeated until the modeled and experimental standard deviations of the residuals are aligned. The results for each iteration of this process are listed in table 5. The standard deviations include up to two decimal places for alignment purposes only. This level of precision is not required for the final measurement noise variance.

$R(t_k)$	$\sigma_{r(t_k),actual}$	$\sigma_{r(t_k),KF}$
2×10^{-21}	1.29×10^{-10}	1.17×10^{-10}
3.5×10^{-21}	1.24×10^{-10}	1.27×10^{-10}
3×10^{-21}	1.26×10^{-10}	1.24×10^{-10}
3.2×10^{-21}	1.25×10^{-10}	1.25×10^{-10}

Table 5: Measurement Noise Variance - Aligning the Actual and Modeled Std. Deviations of the Residuals

A measurement noise variance of 3.2×10^{-21} in the fractional frequency better characterizes the Ettus N310 measurement system. Now, the updated KF jump detection block successfully decreased the number of false alarms during program execution. There is one 'jump' detected associated with the first fractional frequency computed upon program initialization. This is a result of the fractional frequency data itself having a nonzero mean (see Fig. 32). Initially, the KF jump detection algorithm assumes the fractional frequency is centered around zero. After the first correction step, the expected fractional frequency is closer to the actual mean value. To resolve this false alarm, the measurement noise variance can be increased or the initial output of the KF jump detection block can be ignored. Figure 33 depicts the residuals of the fractional frequency data in figure 32. The jump threshold set by the KF jump detection block is included for reference.

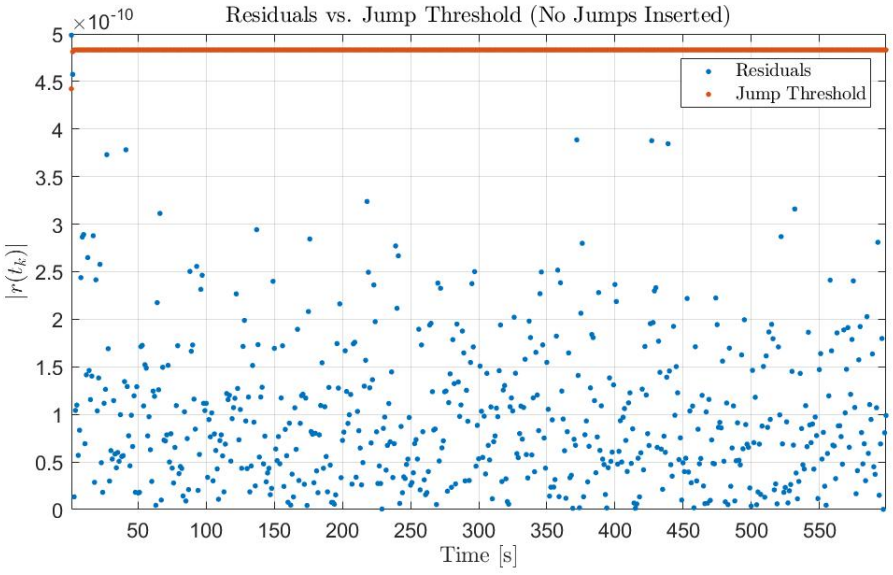


Figure 33: Residual Detector - Residuals vs. Jump Threshold

Clearly, all of the residuals lie within the bounds of the jump threshold, except for one. To reiterate, the first fractional frequency value computed upon program initialization leads

to a relatively large residual. To verify the functionality of the detection system, more tests must be performed using different clocks. Note that the noise in the fractional frequency is partly dependent on the length of the averaging window. Therefore, the measurement noise variance may need to be adjusted for different window sizes. To address this, the measurement noise variance was made an input to the KF jump detection block within GNU radio. The user should set this value to match the specific test configuration.

9.2 Error Injection

The clock error detection algorithm is implemented in GNU radio. In order to test the functionality of the detection system, there must be errors present in the CSAC clock signals. There are a few options available to test the detection system other which do not require waiting for these clock errors to occur naturally. First, the faults can be induced. That is, the CSACs could be subjected to adverse conditions (e.g. extreme temperatures) which may induce a clock error. However, this approach requires purposefully degrading the performance of these expensive clocks. Instead, these clock errors can be inserted onto the CSAC signals in simulation. To be specific, the digitally sample CSAC signals can be modified within GNU radio upstream of the detection system.

The first type of clock error considered is a frequency jump. The CSACs are designed to output a stable 10MHz signal. Frequency jumps appear as sudden discontinuities in this CSAC signal frequency, differentiating them from frequency drift which occurs over a longer duration of time. So, injecting a frequency jump requires instantaneously increasing/decreasing the frequency of the sampled signal.

This is achieved using a single side-band (SSB) complex mixer within GNU radio. This SSB mixer makes use of Euler's formula to alter the frequency of the incoming CSAC signal (Eq. 12). The USRP Signal Source block outputs the sampled CSAC as a complex signal. This complex signal is mixed with another low frequency complex signal using a Multiply block within GNU radio. The resulting complex signal is the downconverted CSAC clock offset by a finite frequency, f_{jump} (Eq. 13).

$$e^{ix} = \cos(x) + i\sin(x), \quad e^{-ix} = \cos(x) - i\sin(x) \tag{12}$$

$$e^{i2\pi f_{CSAC}} \cdot e^{\pm i2\pi f_{jump}} = e^{i2\pi(f_{CSAC} \pm f_{jump})} \tag{13}$$

The low frequency 'jump' signal, $e^{i2\pi f_{jump}}$, is generated using two real Signal Source blocks in GNU radio. One signal is set as a cosine waveform, while the other is set as a sine waveform. The frequency is set by the user during program execution via a GUI. These

two real signals are combined into a single complex signal using the Float to Complex block in GNU radio (Eq. 12). When a positive frequency jump is desired, the sine waveform is directly input to the imaginary component port of the Float to Complex block. When a negative frequency jump is desired, the sine and cosine waveforms are input to the opposite ports of the Float to Complex block. The magnitude and direction of the frequency jumps are controlled by the user during program execution through the GUI.

The second type of clock error considered for the fault injection system this semester was the phase jump. Phase jumps are a common type of clock error, where there is a significant discontinuity between the measured and expected phases at a given sample epoch. If the unwrapped phase of a signal is given by Eq. 14, then a phase jump at a time t_1 is represented by Eq. 15. Note that the frequency of the signal is unaffected. The clock error only appears in the phase data. Nevertheless, the phase jump will appear in the average fractional frequency and is detectable using the KF jump detection block described in the previous section.

$$\theta(t) = 2\pi f t \tag{14}$$

$$\theta(t) = \begin{cases} 2\pi f t & t \leq t_1 \\ 2\pi f t + \delta\theta & t > t_1 \end{cases} \tag{15}$$

So, inserting phase jumps requires adding a constant phase $\delta\theta$ shift for all time after the jump epoch t_1. Within GNU radio, this is achieved by adding a constant angle offset (in radians) to the measured signal phase. The complex signal output by the SSB mixer is converted to wrapped phase data through a Complex to Arg block in GNU radio. The constant angle offset is added to this wrapped phase data. The unwrapped phase and average fractional frequency are computed next, before being input to the KF jump detection block. The magnitude and direction of these phase jumps are controlled by the user through the GUI during program execution.

9.3 Pseudo Real-Time Experiments

The tools have been developed to inject two different types of clock errors. In addition, a detection algorithm has been implemented in GNU radio to flag the clock errors during live tests. The full error and detection grc flowgraph used for testing is depicted in figures 34 and 35, below.

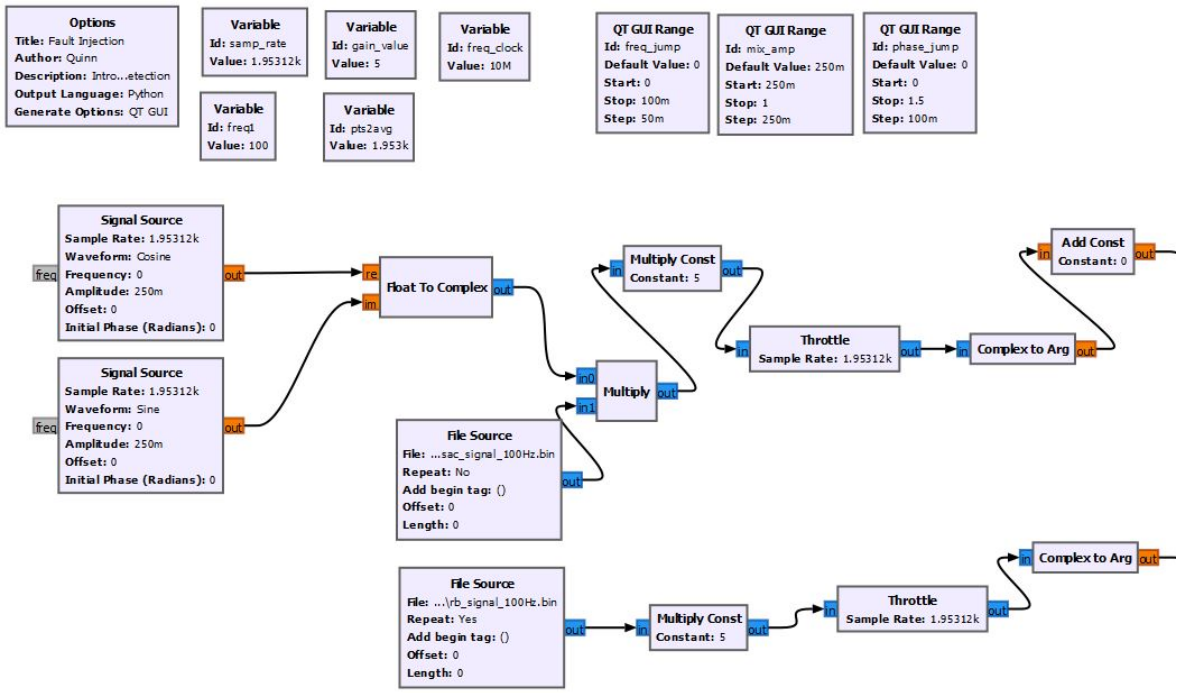


Figure 34: Clock Error Injection/Detection GRC (1/2)

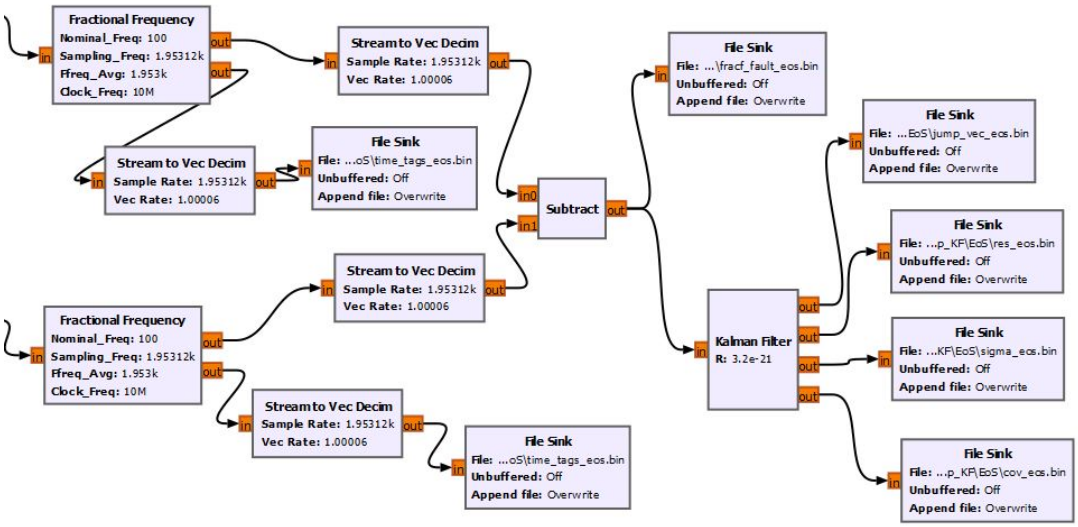


Figure 35: Clock Error Injection/Detection GRC (2/2)

When executed, the program opens up a GUI where the user can control the frequency and phase jumps inserted onto the CSAC signal. The GUI is depicted in figure 36.

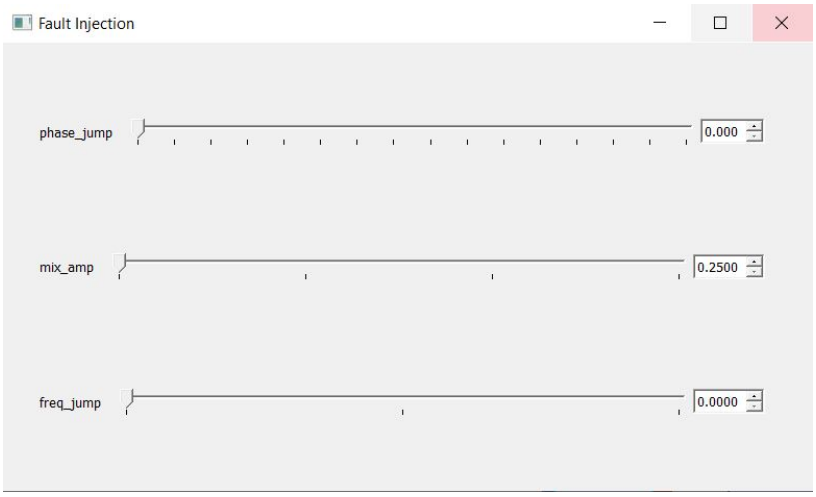


Figure 36: Clock Error Injection/Detection GUI

Pseudo real-time experiments were conducted using recorded CSAC and Rb signal measurements. These complex signals were read from binary files at the same sampling rate at which they were recorded. In one experiment, two frequency jumps and two phase jumps were injected onto a CSAC signal. Both frequency jumps were 50mHz in magnitude, but in opposite directions. Similarly, both phase jumps were 0.1rad in magnitude, but in opposite directions. The average fractional frequencies of the 'faulty' CSAC signal are computed with respect to a Rb reference clock and input to the KF jump detection block.

Figure 37 depicts the average fractional frequencies computed during the pseudo real-time experiment. The two frequency jumps appear as step discontinuities in the fractional frequency data. The two phase jumps appear as two spikes in the fractional frequency data. All four injected clock errors are labeled in the top plot. Next, the output of the KF jump detection block are inspected. The bottom plot of figure 37 portrays the magnitudes of the residuals (Eq. 10) as well as the the jump threshold. There are four distinct intervals where the residuals are above the jump threshold. The KF jump detection block flags these indices to indicate the presence of jumps. Importantly, the detected errors are in alignment with the injected frequency and phase jumps (i.e. with respect to time).

Next, the connection between the two plots is studied. Each frequency jump produces three residuals above the jump threshold. The first residual corresponds to the averaging window in which the frequency jump occurs. The frequency jump produces a change in the fractional frequency during this window. The change is large enough to register a jump. However, the fractional frequency has not settled at the mean value corresponding to the new signal frequency $f_{CSAC} \pm f_{jump}$ (see the intermediate fractional frequencies between the two distinct levels). Consequently, the KF fractional frequency estimate will not match the subsequent measurement, and will likely result in another registered jump. The following

average fractional frequency corresponds to the new signal frequency. The KF estimate will update again. Finally, if the new estimate is still too far away, one more jump is registered as the KF begins to track the new signal frequency. In the end, the first two residuals following the frequency jump will be relatively large. The third residual will be much closer to, or even below, the jump threshold.

The phase jumps produce two residuals above the jump threshold. The first residual corresponds to the average fractional frequency computed over the window in which the phase jump occurs. The additional phase deviation results in a fractional frequency far enough away from the KF estimate to register a jump. The KF estimate is updated as it tracks the measured fractional frequencies. However, this phase deviation does not appear in the subsequent averaging window. That is, the subsequent fractional frequency returns to the original level prior to the phase jump. This produces the 'spikes' in the top plot of figure 37. The updated fractional frequency estimate might be too far from the pre-jump value. Thus, another jump is registered and the KF estimate eventually returns to the pre-jump level.

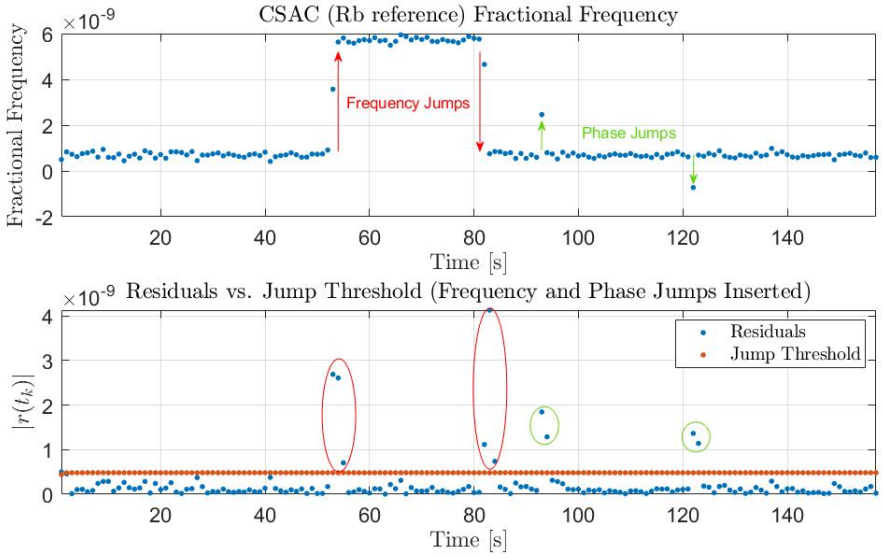


Figure 37: Clock Error Injection and Detection

The error injection system is capable of inserting frequency jumps and phase jumps, which the error detection system effectively identifies and flags.

9.4 Future Work

There are several opportunities for future work with the fault injection and detection systems. First, the fault detection system must be improved. At this point, the detection algorithm

is not capable of distinguishing between different types of clock errors. The error detection system must be able to identify which type of error has occurred in order for the correct error management procedure to be applied. Moreover, the fault injection system is not capable of attributing the clock errors to the test clock or reference clock. The clock ensemble testbed must have the ability to identify which of the individual clocks has experienced errors in order to apply the correct management strategies.

Secondly, the fault injection system must undergo verification testing. To this point, only pseudo real-time experiments have been conducted. The fault injection and detection system should be run with multiple live CSAC signal inputs to mimic the integrated clock ensemble testbed. It will be important to verify whether the fault injection and detection algorithms can keep pace with the sampling rate of the measurement system.

Finally, there is still work to be done defining the limitations of the subsystems. That is, further testing is required to accurately define the range of magnitudes of clock errors which can injected and detected. In addition, the maximum rates at which the subsystems can operate should be identified.

10 Acknowledgements

The team would like to acknowledge Dr. Joanna Hinks and her colleagues at AFRL that attended our presentations and gave helpful feedback for moving forward with the project. Additionally, the team would like to acknowledge Dr. Nicholas Rainville for providing project guidance, and Anastasia Muszynski from MAXWELL for her important contributions on the orbital CSAC experiment. The CONTACT team would like to give special recognition to Dr. Penina Axelrad for her invaluable support during this unique semester.

References

[1] Strategy for the department of defense positioning, navigation, and timing (pnt) enterprise. Technical report, 2019.

[2] H. Aliakbarian, V. Volski, E. van der Westhuizen, R. Wolhuter, and G. A. E. Vandenbosch. Analogue versus digital for baseband beam steerable array used for leo satellite applications. In *Proceedings of the Fourth European Conference on Antennas and Propagation*, pages 1–4, 2010.

[3] F. Kearny D. Frizelle. *Complex RF Mixers, Zero-IF Architecture, and Advanced Algorithms: The Black Magic in Next-Generation SDR Transceivers.* Analog Devices, February 2017.

[4] Peter H. Dana and B. Penrod. The role of gps in precise time and frequency dissemination. 2001.

[5] I. Diegel. Hydrogen maser frequency standard. presented at Seventh IVS Technical Operations Workshop, 2013.

[6] Marion Gödel and Johann Furthner. Robust ensemble time onboard a satellite. 03 2017.

[7] Lee Mohon. Deep space atomic clock (dsac) overview.

[8] National Instruments. *NI myRIO-1900 User Guide and Specifications.* 2016.

[9] J. Sherman and R. Jordens. Oscillator metrology with software defined radio. *Review of Scientific Instruments*, 87(054711):1–11, 2016.

11 Appendix