

UNIVERSITY OF OKLAHOMA
GRADUATE COLLEGE

DIGITAL SYNCHRONIZATION OF SOFTWARE-DEFINED RADIOS
IN A WIRELESS DISTRIBUTED RADAR NETWORK

A THESIS
SUBMITTED TO THE GRADUATE FACULTY
in partial fulfillment of the requirements for the
Degree of
MASTER OF SCIENCE

By
Cody Kieu
Norman, Oklahoma
2025

SYNCHRONIZATION OF SOFTWARE-DEFINED RADIOS
IN A DISTRIBUTED NETWORK

A THESIS APPROVED FOR THE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY THE COMMITTEE CONSISTING OF

Dr. Russell Kenney, Chair

Dr. Nathan Goodman

Dr. Justin Metcalf

Dedicated to my Mom and Dad

Acknowledgments

With the greatest honor, I acknowledge my advisor and friend, Dr. Russell Kenney. I would never have had the opportunity to grow academically to this extent without him. Dr. Kenney showed me that while research can be demanding, it can also be rewarding, and I thank him for making this journey enjoyable. Similarly, I would like to thank my committee members, Dr. Nathan Goodman and Dr. Justin Metcalf, for providing me with invaluable insight over these past few years. The many hours I spent discussing with them have greatly deepened my understanding of radar signal processing.

I would also like to thank Shane Flandermeyer and Rylee Mattingly for teaching me everything I need to know about Linux and GNU Radio. Without them, I would not have been able to conduct my research to this extent.

Then, through the highs and lows of this journey, I would have surrendered to the stresses of research if not for the support of my closest friends, Breanna, Lucía, Bryce, and Sam. The countless crosswords, Guitar Hero sessions, and spontaneous sweet treat trips have made this adventure way less sad and infinitely more fun.

Finally, I would like to thank my family, especially my Mom, Dad, sister, and our dog Luna, for being my foundation and supporting me throughout my life. Your unconditional love and patience have helped me become a better person, and for that, I am deeply grateful. They have provided everything I ever needed to get me to succeed in all aspects of my life. I will love you forever and a day.

This work was funded by the Office of Naval Research.

Table of Contents

Acknowledgments	v
Table of Contents	viii
List of Tables	ix
List of Figures	xii
Abstract	xiii
1 Introduction	1
1.1 Motivation	2
1.2 Contributions	4
1.3 Outline	4
2 Background	6
2.1 Radar Signal Modeling and Synchronization Framework	6
2.2 Synchronization Algorithm Derivation	9
2.2.1 Time-Division Multiple Access Model	10
2.2.2 Estimation of Clock Drift	11
2.2.2.1 Single-Tone Based Frequency Synchronization	12
2.2.2.2 Estimation of the Tone Frequency Using Sinc Non- Linear Least Squares	14
2.2.2.3 Ordinary Least-Squares	16

2.2.3	Clock Drift Compensation	17
2.2.4	Estimation of Clock Bias and Carrier Phase	19
2.2.4.1	Matched Filtering	20
2.2.5	Clock Bias and Carrier Phase Compensation	22
2.2.6	Fully Synchronized Signal Model	23
2.3	Software Defined Radio	24
2.3.1	Universal Software Radio Peripheral	25
2.4	GNU Radio	27
2.4.1	GNU Radio Architecture	29
2.4.2	GNU Radio Implementation of Pulsed Radar via the Message Do- main	30
2.4.3	Fast Fourier Transform-Based Fractional Delay	33
3	Implementation of Synchronization Algorithm with USRP and GNU Radio	34
3.1	Objectives	34
3.2	gr-harmonia: A Custom Out-of-Tree Module for SDR Synchronization . . .	35
3.2.1	Waveform Generation	35
3.2.2	USRP Interfacing and Control	39
3.2.3	Waveform Compensation	44
3.2.4	Clock Drift Estimation Blocks	45
3.2.5	Clock Bias and Carrier Phase Estimation Blocks	48
3.3	Flowgraph Implementation and Module Limitations	54
4	Experimentation and Analysis	58
4.1	Hardware Setup and Calibrations	58
4.1.1	Hardware Setup	59
4.1.2	Hardware Calibration	60
4.2	Experimental Setups and Performance Analysis	61
4.2.1	Cramér-Rao Lower Bound	62

4.2.2	Clock Drift Estimation in A Cabled Two-Node System	70
4.2.3	Clock Bias and Carrier Phase Estimation in Over-the-Air System Experiments	72
5	Conclusions and Future Work	78
5.1	Conclusions	78
5.2	Future Work	79
	References	81
	Appendix A Acronyms	86

List of Tables

3.1	Possible Metadata Keys for an Output PDU of the Single-Tone Block. . . .	37
3.2	Possible Metadata Keys for an Output PDU of the LFM Block.	40
3.3	Metadata Key for an Output PDU of the UHD Block.	44
3.4	Metadata Keys for an Output PDU of the Frequency Peak Estimation Block.	46
3.5	Metadata Keys for an Output PDU of the Clock Drift Estimation Block. . .	48
3.6	Metadata Keys for an Output PDU of the Peak Time Estimation Block. . . .	51
3.7	Metadata Keys for an Output PDU of the Clock Bias and Carrier Phase Estimation Block.	53
4.1	Important RF specification of the Ettus USRP X310 and UBX-160 daugh- terboard. Data from [24], [30].	60
4.2	Parameters for the Two-Node Clock Drift Estimation Experiment.	71
4.3	Parameters for the Full Over-the-Air Synchronization Experiment.	74
4.4	Clock Drift Estimation Results.	75
4.5	Range and Clock Bias Estimation Results.	76

List of Figures

2.1	Time-division multiple access diagram.	10
2.2	Ideal pulsed LFM signal.	20
2.3	Matched filter response (top) and the output of the matched filter (bottom). .	21
2.4	A general block diagram of USRP architecture. From [23].	26
2.5	Block diagram of the UBX 160 daughterboard architecture. From [24]. . .	27
2.6	Example flowgraph in GNU Radio Companion. From [25].	29
2.7	Block diagram of UHD and its role between GNU Radio and USRP hard- ware. From [23].	31
3.1	Single-Tone Source Block. The block generates a single-tone frequency waveform at baseband.	36
3.2	Example output of a single-tone block.	37
3.3	LFM Source Block. Generates a linearly frequency-modulated waveform at baseband.	39
3.4	Example waveform output of an LFM block.	39
3.5	UHD: All USRP Radars Block. Controls all signal exchanges between a three-SDR network and the configuration of each radar.	41
3.6	Waveform Compensation Block. The block adjusts received waveforms with clock drift, clock bias, and carrier phase estimates.	45
3.7	Frequency Peak Estimation Block. The block estimates the frequency of single-tone signals and assigns them to the corresponding receiving SDR. .	46

3.8	Clock Drift Estimation Block. Estimates the clock drift using the frequency estimates from the Frequency Peak Estimation Block.	47
3.9	Flowgraph connection between the Frequency Peak Estimation blocks and the Clock Drift Estimation block.	48
3.10	Time Peak Estimation Block. The block estimates time and phase based on the received pulse compression waveform.	49
3.11	Clock Bias and Carrier Phase Estimation Block. Calculates the clock bias and carrier phase offsets based on the time and phase estimates from the Peak Time Estimation Block.	51
3.12	Flowgraph connection between the Time Peak Estimation blocks and the Clock Bias and Carrier Phase Estimation block.	54
3.13	Flowgraph diagram showing how the gr-harmonia blocks are interconnected to perform synchronization in a distributed radar system.	55
4.1	Monopole antenna calibration setup.	61
4.2	Frequency Cramér-Rao Lower Bound with cabled experiment parameters. .	63
4.3	Frequency Cramér-Rao Lower Bound with OTA experiment parameters. . .	64
4.4	Clock drift Cramér-Rao Lower Bound with cabled experiment parameters. .	65
4.5	Clock drift Cramér-Rao Lower Bound with OTA experiment parameters. . .	65
4.6	Time Cramér-Rao Lower Bound.	66
4.7	Clock bias Cramér-Rao Lower Bound.	67
4.8	Range Cramér-Rao Lower Bound.	68
4.9	Phase Cramér-Rao Lower Bound.	68
4.10	Carrier phase Cramér-Rao Lower Bound.	69
4.11	Block diagram of cabled two-node system setup used to assess clock drift estimation.	70
4.12	Two-node cabled system setup for evaluating clock drift estimation performance.	72
4.13	FFT of the received signals from SDR 1 and 2 after clock drift compensation.	73

4.14	Block diagram of over-the-air system synchronization setup used to assess synchronization parameter estimation.	74
4.15	Over-the-air synchronization setup in an anechoic chamber for evaluating synchronization parameter estimation.	76
4.16	Over-the-air beamforming experiment in which SDR 1 transmits individually, then SDRs 1 and 2 transmit together, and finally SDR 2 transmits individually to SDR 3 to assess phase coherency.	77

Abstract

Precise synchronization of time, frequency, and phase is crucial for coherent processing in distributed radar systems, especially at higher frequencies where timing errors rapidly degrade performance. To enable distributed radar operations, such as coherent transmit beamforming, distributed radar systems must achieve time synchronization on the order of tens of picoseconds, requiring extraordinarily robust techniques to correct timing errors. The `gr-harmonia` out-of-tree module implements a synchronization algorithm to achieve time, frequency, and phase alignment across distributed radar networks using Universal Software Radio Peripheral (USRP) Hardware Driver (UHD)-compatible software-defined radio (SDR) systems. The algorithm is realized entirely in digital signal processing and thus requires no additional hardware, allowing it to be implemented in networks of unmodified SDRs. Because the synchronization approach is decentralized, it scales efficiently with an arbitrary number of nodes to be synchronized. The module operates exclusively through the GNU Radio message passing interface, converting pulsed data into a protocol data unit (PDU) to reduce buffering-induced latency. The module utilizes UHD burst mode with metadata flags to schedule pulses with accurate timing. It comprises blocks for waveform generation, transmission, and reception, as well as frequency and time peak estimation, clock drift estimation, clock bias, and phase estimation. In this paper, the synchronization framework and functionality of each block are discussed, and the results of the synchronization process are shown for data collected from an open-air test with three Ettus X310s using `gr-harmonia`.

Chapter 1

Introduction

Distributed radar systems, also known as distributed multi-input multi-output (MIMO) radar systems, enable each platform to transmit unique signals and receive incoming signals independently. The development of distributed MIMO radar systems is a growing area of interest due to the versatility and benefits they offer in comparison to traditional monostatic radars [1]. One significant advantage of implementing distributed MIMO radar systems is that they reduce overall costs while providing results comparable to traditional monostatic radars [2]. Additionally, these systems can operate at lower transmit power levels, which in turn reduces the likelihood of detection by external systems [3]. Another benefit is that they lack a single point of failure, resulting in a more stable and robust radar network that can continue to function even if individual nodes become compromised [1]. All this being said, these systems can greatly improve radar capabilities and functionality.

The implementation of distributed MIMO radar systems has led to significant advancements in radar development. For instance, distributed radars have been shown to localize targets with greater accuracy [4]. The distinct vantage points of each node in the distributed radar systems enable them to resolve target velocity that is otherwise affected by Doppler aliasing, which tends to limit the capabilities of single-platform systems [5]. Furthermore, distributed MIMO radar systems have also been shown to enhance power gain, improve beam directivity, and increase detection performance in clutter environments where back-

ground noise and signal interference pose significant challenges [6]. However, these advantages require precise coordination across multiple radar nodes, highlighting the need for software-defined radios (SDRs), which provide the flexibility to implement such systems. The following section discusses the challenges associated with synchronization.

1.1 Motivation

One of the core challenges in building distributed MIMO radar systems is the lack of a direct physical connection between sensors. While the benefits of these systems are significant, the distance between the radars reduces the practicality of having a physical connection between each sensor in the network. As a result, synchronization and coordination must be achieved over-the-air (OTA), which significantly affects the accuracy of information exchange between sensors. To enable reliable communication, the systems require precise localization and synchronization to correct the radars' positioning and ensure the alignment of the radio frequency (RF) electrical state for the radars, specifically their time, frequency, and phase.

Accurately determining each radar's position, also known as localization, is essential in cases where the network is mobile [7]. This is important for the coherent processing of information between nodes to correct for the phase of the incoming signals. Without such correction, images produced by the radars will blur, making targets difficult to detect. In recent studies, higher frequencies, such as the Ka-band and the W-band, have been researched for SAR imaging in automotive radar systems [8, 9]. As a result, the positioning of each radar must be estimated within centimeters or millimeters. Ongoing research focuses on improving navigation estimation in networks through Kalman filtering techniques to enhance performance and stability [10].

Synchronization, as previously noted, is necessary for aligning time, frequency, and

phase across all radar nodes. This alignment becomes especially critical in distributed MIMO radar systems, where high-resolution performance depends on precise coordination between platforms. To accurately compute range, timing offsets between nodes must be compensated to sub-nanosecond accuracy. Without this level of precision, even small timing discrepancies can introduce significant range errors. In addition to timing, mismatches in frequency can result in improper down conversion, corrupting the extraction of the base-band signal and degrading overall signal integrity. Furthermore, if the phases between platforms are not properly aligned, coherent integration becomes difficult or impossible. Instead of combining constructively, the signals may destructively interfere with one another, ultimately leading to the loss of information that would otherwise enhance system performance. Accordingly, achieving synchronization will be the primary objective of this study, as it serves as the foundation for ensuring coherent operation across a distributed radar system.

Global Positioning System (GPS) is commonly used to assist with synchronization, as it provides pulse-per-second (PPS) signals for timing and frequency alignment. In practice, however, GPS is not accurate enough to meet the demanding synchronization requirements of distributed radar systems. Its limitations become clear when considering that its timing accuracy has a standard deviation of approximately 15 nanoseconds, which can introduce unacceptable range errors at high frequencies [11]. In addition to this timing limitation, GPS does not inherently provide phase alignment between nodes, making it insufficient for achieving full synchronization across a distributed network. Furthermore, in certain environments, such as indoor or contested areas, GPS signals may be blocked, jammed, or spoofed, rendering them unreliable. In GPS-denied environments, no practical external timing reference exists, making synchronization across radar nodes extremely challenging.

Due to these challenges, new synchronization techniques are necessary to maintain alignment in time, frequency, and phase across platforms without relying on GPS. Syn-

chronization is ultimately what makes distributed radar systems viable in contested or GPS-denied environments. Although several synchronization methods have been explored in previous research [12–14], many require additional hardware or physical modifications to this platform. This work focuses on the implementation of a fully digital synchronization algorithm developed by Dr. Russell Kenney [15], which can be applied to existing distributed radar systems without any physical modifications.

1.2 Contributions

The thesis makes the following contributions:

- Development of an out-of-tree (OOT) module, *gr-harmonia*, for GNU Radio, which includes custom signal processing blocks designed for software-defined radios (SDR) network synchronization.
- Implementation of a multi-burst time-division multiple access (TDMA) scheme, waveform generation, and parameter estimation using metadata-tagged transmission across distributed Universal Software Radio Peripheral (USRP) platforms.
- Evaluation of synchronization performance under stationary conditions using over-the-air (OTA) data.

1.3 Outline

The document is organized as follows:

Chapter 2 outlines the synchronization algorithm and the underlying estimation techniques, including nonlinear least squares (NL-LS), ordinary least squares (OLS), and matched filtering. Additionally, it explains the use of time-division multiple access (TDMA) schemes for estimating and compensating clock drift, clock bias, and carrier

phase. It also introduces GNU Radio, software-defined radios (SDRs), and Universal Software Radio Peripherals (USRPs), providing an in-depth explanation of their architectures and how SDRs interface within the GNU Radio framework. In addition, it discusses message-domain communication, protocol data units (PDUs), and their relevance to pulse radar applications.

Chapter 3 explains the implementation of the synchronization algorithm within the `gr-harmonia` module. It describes how the algorithm is divided into modular GNU Radio blocks that perform tasks, such as waveform generation, estimation, and compensation of key synchronization components. Additionally, it discusses how these blocks exchange PDUs and signals within the message domain to coordinate transmission and estimation. Finally, it highlights the integration of the GNU Radio blocks to form a complete flowgraph designed to synchronize a network of three SDRs.

Chapter 4 presents the experimental setup and results, describing the two-SDR loop-back configuration and the three-SDR over-the-air (OTA) tests. This chapter conducts an analysis of the synchronization algorithm and a comparison to the Cramer-Rao Lower Bound (CRLB) to evaluate the accuracy and limitations of the module.

Chapter 5 summarizes the key findings and discusses the implications for distributed MIMO radar synchronization. It also highlights potential directions for future research, including improving estimation accuracy and platform motion.

Chapter 2

Background

This chapter outlines the signal processing techniques, hardware components, and software frameworks involved in the design and evaluation of this research. It begins by introducing the signal model and the key sources of signal misalignment in distributed radar systems: clock drift, clock bias, and carrier phase offsets. Next, the TDMA scheme is discussed for the coordinated transmission required for the distributed radar platforms to operate without interfering with one another. Following that, the synchronization algorithm is introduced, detailing how clock drift, clock bias, and carrier phase are estimated and compensated for across the radar system. Lastly, the foundations of software-defined radios (SDRs) and GNU Radio are presented to provide an understanding of the hardware and software used during experimentation.

2.1 Radar Signal Modeling and Synchronization Framework

The signal model used in this work follows the algorithm presented in [15]. Several assumptions are made to simplify the signal modeling of a distributed radar network. First, it is assumed that the network consists of spatially distributed radar sensors, each equipped with an independent main oscillator (MO). These oscillators are assumed to operate at the same nominal frequency across all platforms. Each platform uses its MO to drive both the

digital sampling clocks and the local oscillator (LO). This allows any frequency errors in a platform's MO to produce a proportional offset in all timing and frequency-dependent operations throughout the system.

The nominal frequency of each MO is given by f_{MO} . Due to the imperfect nature of crystal oscillators, the true frequency of the MO on platform i will deviate slightly from the nominal frequency, where the true frequency can be denoted by $f_{\text{MO},i}$. Now that true frequency and nominal frequency are introduced, *clock drift* can be defined as the ratio of the two, given by

$$\alpha_i \triangleq \frac{f_{\text{MO},i}}{f_{\text{MO}}}. \quad (2.1)$$

Additionally, each platform is assumed to have a random but constant *clock bias*, denoted by ϕ_i . This bias can be attributed to multiple factors, such as the differing initialization time of each platform and transmission chain delays. The local time given from the clock of platform i (τ_i), when written in terms of global time (t), clock drift, and clock bias are given by

$$\tau_i = \alpha_i t + \phi_i. \quad (2.2)$$

Rearranging (2.2) to solve for global time for platform i gives

$$t = \frac{\tau_i - \phi_i}{\alpha_i}. \quad (2.3)$$

A few additional assumptions are made to simplify the signal modeling framework. Each platform should transmit with a nominal carrier frequency of f_c and operate with a nominal sampling frequency f_s for both the digital-to-analog (DAC) and analog-to-digital (ADC) conversions. Moreover, it is assumed that the transmitter and receiver have independent and random phase offset terms, which are given by γ_i^{tx} for the transmitter and γ_i^{rx} for the receiver.

For this project, it is also assumed that the radar platforms are stationary during the synchronization process, removing the need to compensate for platform motion. Let $s_j(t)$ be an arbitrary complex baseband waveform transmitted by platform j . Then it is also assumed platform i has a distance, $R_{i,j}$, from platform j at time t . Because all platforms are stationary, the distance between platforms is constant with respect to time, and by using symmetry, it can be assumed that $R_{i,j} = R_{j,i}$. Dividing the distance by the speed of light gives the signal time-of-flight (TOF) given by

$$TOF_{i,j} = \frac{R_{i,j}}{c}. \quad (2.4)$$

In addition, the difference between the random transmit carrier phase of platform j and the random receive carrier phase of platform i can be represented by

$$\gamma_{i,j}^{\text{err}} = \gamma_j^{\text{tx}} - \gamma_i^{\text{rx}}. \quad (2.5)$$

With these definitions in place, the signal received by platform i from the transmitted signal of platform j in the global time can be illustrated by the equation below

$$\begin{aligned} r_{i,j}(t) = & s_j \left(\alpha_j t + \phi_j - \alpha_j \frac{R_{i,j}}{c} \right) \\ & \cdot \exp \left(j 2\pi f_c \left((\alpha_j - \alpha_i) t + \phi_j - \phi_i - \alpha_j \frac{R_{i,j}}{c} \right) \right) \\ & \cdot \exp(j\gamma_{i,j}^{\text{err}}). \end{aligned} \quad (2.6)$$

By substituting (2.3) into (2.6), the received time of the transmitted signal of platform j

observed in the local clock frame of platform i is shown by

$$\begin{aligned}
r_{i,j}(\tau_i) &= s_j \left(\frac{\alpha_j}{\alpha_i} (\tau_i - \phi_i) + \phi_j - \alpha_j \frac{R_{i,j}}{c} \right) \\
&\quad \cdot \exp \left(j 2\pi f_c \left(\frac{\alpha_j - \alpha_i}{\alpha_i} (\tau_i - \phi_i) + \phi_j - \alpha_j \frac{R_{i,j}}{c} \right) \right) \\
&\quad \cdot \exp(j\gamma_{i,j}^{\text{err}}) \\
&= s_j \left(\frac{\alpha_j}{\alpha_i} \tau_i - \rho \right) \exp \left(j 2\pi f_c \left(\left(\frac{\alpha_j}{\alpha_i} \right) \tau_i \right) \right) \exp(j 2\pi f_c \rho) \exp(j\gamma_{i,j}^{\text{err}}),
\end{aligned} \tag{2.7}$$

where ρ is a substitution variable defined by

$$\rho = \frac{\alpha_j}{\alpha_i} \phi_i - \phi_j + \alpha_j \frac{R_{i,j}}{c}. \tag{2.8}$$

The full derivation of the signal model, including the moving platform case, can be found in [15–17].

2.2 Synchronization Algorithm Derivation

Before presenting the derivation of the synchronization algorithm, it is assumed that a coarse level of synchronization has already been achieved across all radar platforms. This synchronization must be sufficiently accurate to ensure that the transmit and receive windows of all nodes are properly aligned in time within a couple of milliseconds. Such alignment guarantees that each sensor captures the intended transmissions as dictated by the network's scheduling and orthogonality scheme, time-division multiple access (TDMA). This communication scheme is crucial for the fine synchronization process discussed later in this section.

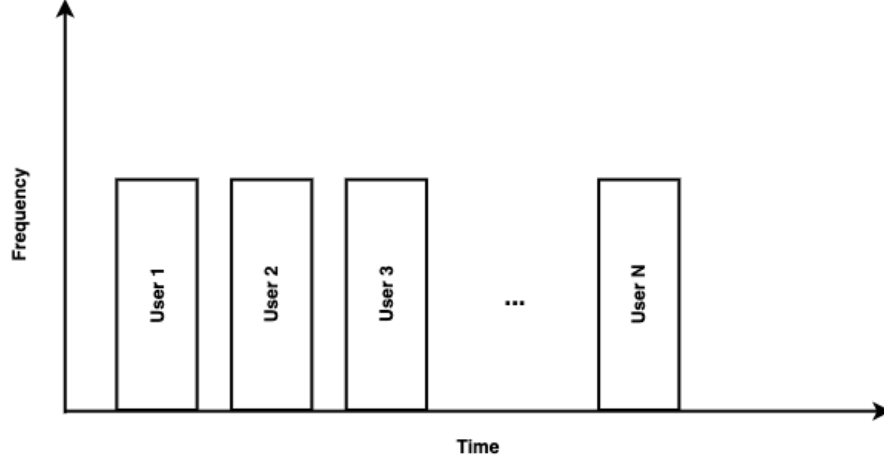


Figure 2.1: Time-division multiple access diagram.

2.2.1 Time-Division Multiple Access Model

Time-division multiple access is a channel access method used in wireless and radar systems to enable multiple devices to share a common communication medium without interfering with one another, in this case, the radar platforms [18, 19]. It works by allocating distinct, non-overlapping time intervals (or slots) to each transmitter, allowing them to operate in a coordinated manner, as illustrated in Figure 2.1. This technique is particularly effective in distributed radar systems where frequency reuse and coordination are essential.

In a TDMA system, time is divided into fixed-length slots of duration Δ_{TDMA} , with each slot assigned to a different node in the network [18]. Suppose there are N total nodes in the system. Each node is assigned a unique integer index 1 to N , which determines its designated transmit time slot. If the nodes were to communicate in ascending order according to their assigned integer values, then the transmit time for node j , relative to its own clock time, is given by

$$\tau_j^{\text{tx}} = (j - 1)\Delta_{\text{TDMA}}, \quad j = 1, 2, \dots, N. \quad (2.9)$$

This indexing scheme ensures that each node transmits during a distinct interval, preserving orthogonality across the TDMA frame. In the global time frame, the transmit time can be given by

$$t_j^{\text{tx}} = (j - 1)\Delta_{\text{TDMA}} - \frac{\phi_j}{\hat{\alpha}_j}, \quad j = 1, 2, \dots, N. \quad (2.10)$$

where $\hat{\alpha}_j$ is the estimate of platform j 's clock drift whose calculations are shown in Subsection 2.2.2.

Following the transmit signal of node j , all other nodes must enter a receive mode to capture the signal. The time at which node i , relative to its local clock, received the signal from node j is

$$\tau_{i,j}^{\text{rx}} = (j - 1)\Delta_{\text{TDMA}} + \frac{\alpha_j}{\hat{\alpha}_j} \frac{R_{i,j}}{c} + \frac{\phi_i}{\hat{\alpha}_i} - \frac{\phi_j}{\hat{\alpha}_j}, \quad (2.11)$$

which in the global time frame can be expressed by

$$t_{i,j}^{\text{rx}} = (j - 1)\Delta_{\text{TDMA}} + \frac{\alpha_j}{\hat{\alpha}_j} \frac{R_{i,j}}{c} - \frac{\phi_j}{\hat{\alpha}_j}. \quad (2.12)$$

Since the TDMA schedule is known throughout all sensors in the network, each sensor can infer the source of any signal.

2.2.2 Estimation of Clock Drift

With the communication scheme established, the discussion now turns to the synchronization algorithm defined in [15]. The first step to synchronizing distributed radar networks is to estimate the clock drift between radar platforms. The following section presents the mathematical formulation used to estimate clock drift based on the measured frequency offsets between platforms. Additionally, techniques for improving frequency estimation to enhance synchronization performance will be discussed.

2.2.2.1 Single-Tone Based Frequency Synchronization

To begin the synchronization algorithm, each platform is designated the same baseband single-tone waveform. Assume that platform j transmits a single-tone at a baseband frequency f_j^α with a pulse duration of T_p . Then, each platform is assigned a transmit time using (2.9). The single-tone baseband transmit signal, expressed in its local time τ_j , is defined as

$$s_j^\alpha(\tau_j) = \exp(j2\pi f_j^\alpha \tau_j) \cdot \text{rect}\left(\frac{\tau_j - \frac{T_p}{2}}{T_p}\right), \quad (2.13)$$

where the rectangular function is defined by

$$\text{rect}(t) \triangleq \begin{cases} 1, & |t| < \frac{1}{2}, \\ 0, & \text{else.} \end{cases} \quad (2.14)$$

The received waveform from platform j to i is then given by

$$\begin{aligned} r_{i,j}^\alpha(\tau_i) = & \exp\left(j2\pi \left[\left(f^c \frac{\alpha_j}{\alpha_i} - 1\right) + \frac{\alpha_j}{\alpha_i} f^{\alpha_j}\right] \tau_i\right) \\ & \cdot \exp(-j2\pi (f^c + f^{\alpha_j}) \rho) \\ & \cdot \text{rect}\left(\frac{\frac{\alpha_j}{\alpha_i} \tau_i - \rho - \frac{T_p}{2}}{T_p}\right) \\ & \cdot \exp(j\gamma_{i,j}^{\text{err}}), \end{aligned} \quad (2.15)$$

where f_c is the center frequency and $\gamma_{i,j}^{\text{err}}$ is the net carrier phase offset defined by (2.5).

The following steps are conducted to estimate the frequency of each received signal. The observed frequency at receiver i can be estimated by computing the Fourier transform

of the received signal, modeled by

$$\begin{aligned}
R_{i,j}^\alpha(f) = & \frac{\alpha_i}{\alpha_j} T_p \operatorname{sinc} \left(\frac{\alpha_i}{\alpha_j} T_p (f - f_{i,j}^\alpha) \right) \\
& \cdot \exp \left[-j2\pi (f - f_{i,j}^\alpha) \left(\frac{\alpha_i}{\alpha_j} \rho + \frac{\alpha_i}{\alpha_j} \frac{T_p}{2} \right) \right] \\
& \cdot \exp \left[-j2\pi (f_c + f_j^\alpha) \rho \right] \\
& \cdot \exp \left(j\gamma_{i,j}^{\text{err}} \right),
\end{aligned} \tag{2.16}$$

where the normalized sinc function is defined as

$$\operatorname{sinc}(x) \triangleq \frac{\sin(\pi x)}{\pi x}, \tag{2.17}$$

and the true frequency of the single tone sent from platform j and received by platform i is given by

$$f_{i,j}^\alpha = \left(\frac{\alpha_j}{\alpha_i} - 1 \right) f_c + \frac{\alpha_j}{\alpha_i} f_j^\alpha. \tag{2.18}$$

The peak of the magnitude of the Fourier transform $|R_{i,j}^\alpha(f)|$ occurs at $f = f_{i,j}^\alpha$, which allows platform i to estimate $\hat{f}_{i,j}^\alpha$ by simply finding the maximum of the received signal.

Once all platforms transmit their single-tone waveforms, platform i has all frequency estimates, $\hat{f}_{i,j}^\alpha$, for each platform such that $j \neq i$. Rearranging (2.18) yields a linear equation of the unknown clock drifts:

$$\left(\hat{f}_{i,j}^\alpha + f_c \right) \alpha_i - \left(f_j^\alpha + f_c \right) \alpha_j = 0. \tag{2.19}$$

This provides $N_p - 1$ independent equations with N_p unknowns, which makes the system undetermined and prevents the clock drifts from being solved. Instead, the values of α_i can be determined relative to one another by selecting one platform's clock drift (e.g.,

α_1) to be set to unity:

$$\alpha_1 = 1. \quad (2.20)$$

The $N_p - 1$ independent equations combined with (2.20) form a fully determined system that allows the clock drifts to be solved. This method is called the *single-tone without exchange technique*. To improve robustness, platforms can also exchange their frequency estimates $\hat{f}_{i,j}^\alpha$ with one another. With this additional information and (2.20), a system of $N_p^2 - N_p + 1$ equations is formed, which allows for a more accurate solution using a least-squares approach. This method is referred to as *single-tone with exchange* and is beneficial in noisy environments. The algorithm in this paper will be implemented using the *single-tone with exchange* method.

2.2.2.2 Estimation of the Tone Frequency Using Sinc Non-Linear Least Squares

Clock drift between independently operating radios introduces small but significant frequency offsets in received signals, even when all devices nominally transmit and receive at the same carrier frequency. To estimate these offsets with high precision, the sinc-based nonlinear least squares (NL-LS) algorithm is used to refine the frequency estimate beyond the resolution of the discrete Fourier transform (DFT). This approach fits a parameterized sinc function to a small range of frequency-domain samples near the spectral peak.

First, assume the received single-tone waveform is processed through a Fast Fourier Transform (FFT) of a time-limited window of duration T_c , yielding a frequency-domain magnitude response. Due to the finite observation time, this response exhibits a sinc-shaped envelope centered at a frequency $f_{\alpha_{i,j}}$ affected by the relative clock drift between radios i and j . The following equation models the spectral magnitude in discrete form

$$|R_{\alpha_{i,j}}[k]| = \left| \frac{\alpha_i}{\alpha_j} T_p \cdot \text{sinc} \left(\frac{\alpha_i}{\alpha_j} T_p T_c (k - T_c f_{i,j}^\alpha) \right) \right|, \quad (2.21)$$

where T_p is the waveform duration, T_c is the capture length, $f_{i,j}^\alpha$ is the drifted frequency, and k is the DFT frequency index.

To refine the frequency estimate, the FFT bin with the highest magnitude is identified, and then its value is interpolated to approximate the true peak of the sinc waveform. This interpolated peak provides the refined frequency estimate. The refinement process begins with the initial peak frequency bin, defined by

$$k_{\text{pk}} = \arg \max_k |R_{\alpha_{i,j}}[k]|. \quad (2.22)$$

The sinc function kernel is given by

$$f_{\text{sinc}}(\mathbf{x}; \boldsymbol{\lambda}) = \lambda_0 \cdot \text{sinc}((x - \lambda_1 \lambda_2)), \quad (2.23)$$

where vector $\mathbf{x}$ is normalized to unit samples expressed by

$$\mathbf{x} = [-1, 0, 1]^T, \quad (2.24)$$

and $\boldsymbol{\lambda}$ controls the parameters of the sinc function, which can be defined as

$$\boldsymbol{\lambda} = [\lambda_0, \lambda_1, \lambda_2]^T. \quad (2.25)$$

The vector $\boldsymbol{\lambda}$ contains the model parameters, where λ_0 is the peak amplitude of the sinc function, λ_1 specifies the frequency value of the peak, and λ_2 controls the width scaling of the sinc function. The vector $\mathbf{y}$ is then defined as the three samples closest to the peak of $|R_{\alpha_{i,j}}[k]|$ given by

$$\mathbf{y} = [|R_{\alpha_{i,j}}[k_{\text{pk}} - 1]|, |R_{\alpha_{i,j}}[k_{\text{pk}}]|, |R_{\alpha_{i,j}}[k_{\text{pk}} + 1]|]. \quad (2.26)$$

The nonlinear least squares cost function to minimize error is given by

$$S(\boldsymbol{\lambda}) = \sum_{i=0}^2 (y_i - f_{\text{sinc}}(x_i; \boldsymbol{\lambda}))^2. \quad (2.27)$$

Using the Gauss–Newton method, we compute updates via the Jacobian matrix $\mathbf{J}$, which consists of:

$$\begin{aligned} \frac{\partial f}{\partial \lambda_0} &= \text{sinc}(\lambda_2(\mathbf{x} - \lambda_1)) \\ \frac{\partial f}{\partial \lambda_1} &= \lambda_0 \cdot \frac{\text{sinc}(\lambda_2(\mathbf{x} - \lambda_1)) - \cos(\pi \lambda_2(\mathbf{x} - \lambda_1))}{\mathbf{x} - \lambda_1} \\ \frac{\partial f}{\partial \lambda_2} &= \lambda_0 \cdot \frac{\cos(\pi \lambda_2(\mathbf{x} - \lambda_1)) - \text{sinc}(\lambda_2(\mathbf{x} - \lambda_1))}{\lambda_2}. \end{aligned} \quad (2.28)$$

Then, to avoid division by zero when $x = \lambda_1$, L'Hôpital's rule is used to compute a stable derivative. At each iteration:

$$\Delta \boldsymbol{\lambda} = (\mathbf{J}^T \mathbf{J})^{-1} \mathbf{J}^T (\mathbf{y} - f(\mathbf{x}; \boldsymbol{\lambda}_n)), \quad \boldsymbol{\lambda}_{n+1} = \boldsymbol{\lambda}_n + \Delta \boldsymbol{\lambda}. \quad (2.29)$$

Once convergence is achieved, the refined tone frequency is computed as

$$\hat{f}_{\alpha_{i,j}} = f[k_{\text{pk}}] + \frac{\tilde{\lambda}_1}{T_c}, \quad (2.30)$$

where $f[k_{\text{pk}}]$ is the peak frequency of the FFT bin, and $\tilde{\lambda}_1$ is the sub-bin correction from the sinc NL-LS solution, given by the vector $\tilde{\boldsymbol{\lambda}}$. This estimate is used to determine the effective frequency offset introduced by clock drift.

2.2.2.3 Ordinary Least-Squares

After using sinc-NLLS to estimate the frequency of each received signal, clock drift can be estimated by solving the set of linear equations. The general system of linear equations can

be represented as

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \quad (2.31)$$

where $\mathbf{A}$ is an $M_b \times M_x$ matrix of coefficients of the system of equations given from (2.19) and that relates vectors $\mathbf{x}$ and $\mathbf{b}$. The vector $\mathbf{x}$ has M_x values to be solved for, and $\mathbf{b}$ is an $M_b \times 1$ solution vector containing the values for the solution of each equation. Assuming $M_b > M_x$, the solution to the system of equations is ordinary least-squares (OLS), which uses the Moore-Penrose pseudo-inverse matrix. The solution is computed using the following formula

$$\hat{\mathbf{x}} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}. \quad (2.32)$$

It is assumed $\mathbf{b}$ is corrupted by additive noise with covariance $\mathbf{C}_b$. If the variance of each component of $\mathbf{b}$ is equivalent, then OLS can be used. Otherwise, if the variance of each component is different, then weighted least-squares (WLS) can be used to improve estimation accuracy [20, 21]; however, for simplicity, OLS estimation is used for the implementation of this algorithm.

2.2.3 Clock Drift Compensation

After clock drift estimation is completed, each platform i has an estimate of its clock drift, denoted by $\hat{\alpha}_i$. With this information, each platform can compensate for its clock drift by modifying the time scaling of its transmitted and received signals to align them with a common reference frame, achieving frequency synchronization. It is essential to note that Doppler shifts resulting from platform motion can affect frequency synchronization; however, they are disregarded in this paper since all platforms are assumed to be stationary.

Starting with the local time-domain transmit signal from platform j , the time dilation due to clock drift α_j can be compensated by reconstructing the baseband signal by divid-

ing the time axis by the estimated clock drift, $\hat{\alpha}_j$. The carrier frequency of the signal is corrected by multiplying the signal by a compensating sinusoid. The resulting corrected transmit baseband waveform $s'_j(\tau_j)$ is given by

$$s'_j(\tau_j) = s_j \left(\frac{\tau_j}{\hat{\alpha}_j} \right) \cdot \exp \left[j2\pi f_c \left(\frac{1}{\hat{\alpha}_j} - 1 \right) \tau_j \right]. \quad (2.33)$$

The received signal at platform i in its local time frame after applying compensation on transmit is given by

$$\begin{aligned} r'_{i,j}(\tau_i) = & s'_j \left(\frac{\alpha_j}{\alpha_i} \tau_i - \rho \right) \cdot \exp \left[j2\pi \left(\left(\frac{\alpha_j}{\hat{\alpha}_j \alpha_i} - 1 \right) f_c \right) \tau_i \right] \\ & \cdot \exp \left[-j2\pi f_c \left(\frac{\rho}{\hat{\alpha}_j} \right) \right] \cdot \exp [j\gamma_{i,j}^{\text{err}}]. \end{aligned} \quad (2.34)$$

Let the corrected receive time axis be defined as

$$\tau'_i = \frac{\tau_i}{\hat{\alpha}_i}. \quad (2.35)$$

Correcting the time axis of the received signal gives the following equation:

$$\begin{aligned} r'_{i,j}(\tau'_i) = & s_j \left(\frac{\alpha_j \hat{\alpha}_i}{\hat{\alpha}_j \alpha_i} \tau'_i - \frac{\rho}{\hat{\alpha}_j} \right) \cdot \exp \left[j2\pi \left(\frac{\alpha_j \hat{\alpha}_i}{\hat{\alpha}_j \alpha_i} - \hat{\alpha}_i \right) f_c \tau'_i \right] \\ & \cdot \exp \left[-j2\pi f_c \left(\frac{\rho}{\hat{\alpha}_j} \right) \right] \cdot \exp [j\gamma_{i,j}^{\text{err}}]. \end{aligned} \quad (2.36)$$

Assuming accurate estimation across platforms, the following can be approximated into

$$\frac{\alpha_j \hat{\alpha}_i}{\hat{\alpha}_j \alpha_i} \approx 1, \quad \frac{\alpha_j}{\hat{\alpha}_j \alpha_i} \approx \frac{1}{\hat{\alpha}_i}. \quad (2.37)$$

So, as a result, the signal becomes

$$r'_{i,j}(\tau'_i) = s_j \left(\tau'_i - \frac{\rho}{\hat{\alpha}_j} \right) \cdot \exp [j2\pi ((1 - \hat{\alpha}_i) f_c) \tau'_i] \cdot \exp \left[-j2\pi f_c \left(\frac{\rho}{\hat{\alpha}_j} \right) \right] \cdot \exp [j\gamma_{i,j}^{\text{err}}] . \quad (2.38)$$

To correct for the carrier frequency offset, signal $b_i(\tau'_i)$ must be multiplied by the received signal. The carrier compensation signal is defined as

$$b_i(\tau'_i) = \exp [j2\pi f_c (\hat{\alpha}_i - 1) \tau'_i] . \quad (2.39)$$

Then the final compensated signal is

$$\begin{aligned} \tilde{r}_{i,j}(\tau'_i) &= r'_{i,j}(\tau'_i) \cdot b_i(\tau'_i) \\ &= s_j \left(\tau'_i - \frac{\rho}{\hat{\alpha}_j} \right) \cdot \exp \left[-j2\pi f_c \left(\frac{\rho}{\hat{\alpha}_j} \right) \right] \cdot \exp [j\gamma_{i,j}^{\text{err}}] . \end{aligned} \quad (2.40)$$

The resulting waveform is free of time dilation and carrier frequency offset, enabling coherent operation among the distributed platforms.

2.2.4 Estimation of Clock Bias and Carrier Phase

To achieve full synchronization, it is necessary to account for the unknown time shifts and phase offsets introduced by initialization times and RF front-end phase variations. The procedure for estimating clock bias and carrier phase is given in [15, 18, 22]. In addition, it is assumed the radar system is frequency-synchronized, which in turn isolates the problem to time and phase alignment.

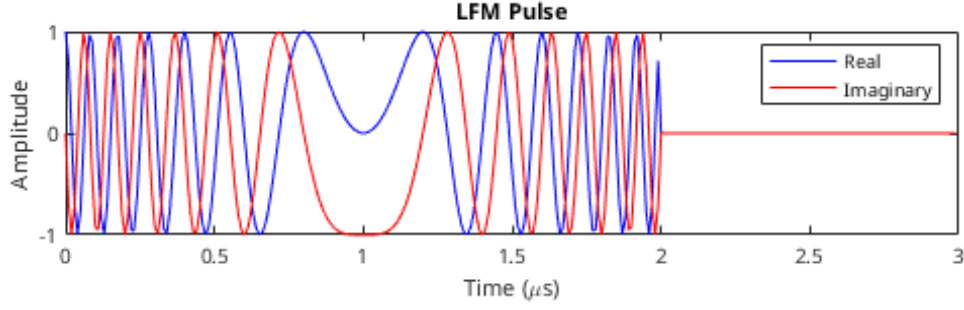


Figure 2.2: Ideal pulsed LFM signal.

2.2.4.1 Matched Filtering

The synchronization of clock bias and carrier phase begins with each platform broadcasting a pulse-compression waveform, which can be used for time-delay estimation. In this paper, the pulse-compression waveform is a linear frequency modulation (LFM) pulse, which can be denoted by

$$x_p(t) = \exp \left[-j2\pi \frac{\beta}{2} t + j\pi \frac{\beta}{T_p} t^2 \right] \text{rect} \left(\frac{t - \frac{T_p}{2}}{T_p} \right), \quad (2.41)$$

where β is the signal bandwidth. A plot of an ideal pulsed LFM in the time-domain with a bandwidth of 25 MHz and a pulse length of 2 μs is shown in Figure 2.2.

Matched filtering, also known as pulse compression, provides the benefit of maximizing the signal-to-noise ratio (SNR) of the input signal, especially in the presence of noise. The matched filter response is built by calculating the conjugate and time-reversal of the transmitted signal, given by

$$h(t) = x_p^*(-t). \quad (2.42)$$

Suppose the received signal, $x(t)$, is matched filtered, and $y(t)$ is the output of passing the received signal through the filter $h(t)$. The matched filtered signal can be written as

$$y(t) = \int_{-\infty}^{\infty} h(\tau - t) x(t) dt = x(t) \otimes h(t), \quad (2.43)$$

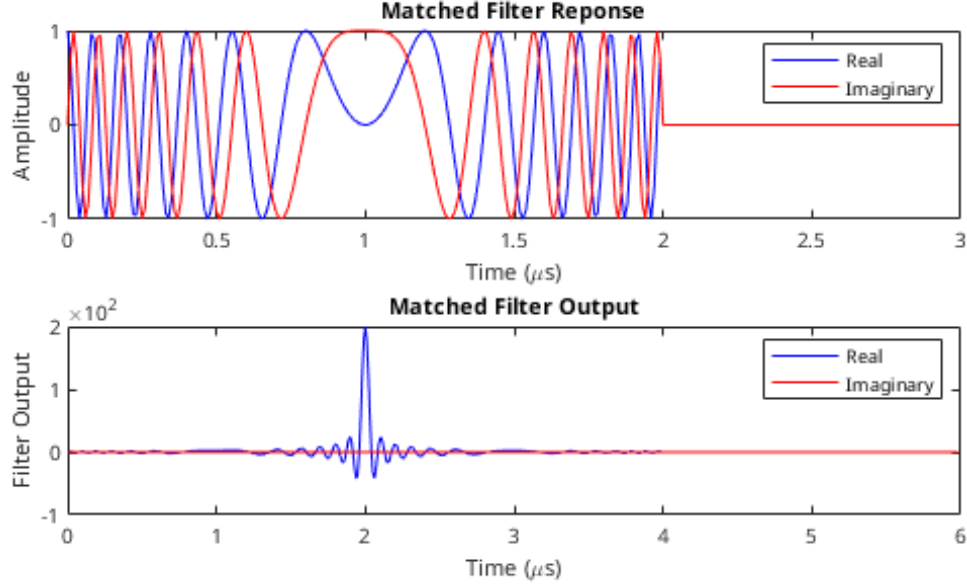


Figure 2.3: Matched filter response (top) and the output of the matched filter (bottom).

where $\otimes$ is the symbol that denotes convolution. The matched filter response and the output of the matched filter of the LFM signal are illustrated in Figure 2.3.

It is important to note that the magnitude of the complex pulse-compression output will always have a sinc function shape. Furthermore, the matched filter process is mathematically equivalent to the cross-correlation between the received signal and the transmitted signal. With the LFM waveform and matched filtering established, clock bias and carrier phase estimation can be addressed.

After each platform j transmits an LFM, all other platforms compute the cross-correlation of the received signal $r_{i,j}^b(\tau'_i)$ with the known baseband LFM waveform $s_j(\tau'_i)$. This then gives each platform the cross-correlation signal $d_{i,j}(\tau'_i)$, from which each platform can compute the peak value of the cross-correlation $m_{i,j}$. However, due to the resolution of the DFT, the time-domain sinc-based NL-LS is used to further improve the time estimation of the peak value of $d_{i,j}$. The only difference in the sinc NL-LS estimation

technique is the initialization vector given by

$$\lambda_0 = \begin{bmatrix} |R_{i,j}[k_{pk}]| & 0 & \frac{\beta}{F_s} \end{bmatrix}. \quad (2.44)$$

The time delay values $m_{i,j}$ determined by platform i for all platforms j , where $j \neq i$, can be expressed by

$$m_{i,j} = \frac{\rho}{\hat{\alpha}_j} = \frac{1}{\hat{\alpha}_i} \phi_i - \frac{1}{\hat{\alpha}_j} \phi_j + \frac{\alpha_j}{\hat{\alpha}_j} \frac{R_{i,j}}{c}. \quad (2.45)$$

Additionally, to help compute carrier phase offsets, the phase of the cross-correlational signal at the peak value $\angle d_{i,j}(m_{i,j})$ should be calculated on each platform, resulting in the following equation:

$$\angle d_{i,j}(m_{i,j}) = \gamma_{i,j}^{err} - 2\pi f_c m_{i,j}. \quad (2.46)$$

Then, using a communication channel (such as BPSK-modulated signals), all platforms exchange their values of $m_{i,j}$ and $\angle d_{i,j}(m_{i,j})$ for all (i, j) pairs where $j \neq i$. Once all values are received, every platform should have the same knowledge of $m_{i,j}$ and $\angle d_{i,j}(m_{i,j})$.

By exploiting the signal symmetry of stationary platforms, the range estimates $\hat{R}_{i,j}$ can be calculated by

$$\hat{R}_{i,j} = \frac{\alpha_j}{\hat{\alpha}_j} R_{i,j} = \frac{c(m_{i,j} + m_{j,i})}{2}. \quad (2.47)$$

Similarly, an estimate of the clock bias differences can then be computed by

$$\hat{\phi}_{i,j} = \frac{1}{\hat{\alpha}_i} \phi_i - \frac{1}{\hat{\alpha}_j} \phi_j = \frac{m_{i,j} - m_{j,i}}{2}. \quad (2.48)$$

2.2.5 Clock Bias and Carrier Phase Compensation

After estimating the clock bias differences, the clock bias of each platform can be calculated to align with the network mean [15]. Using the computed clock bias differences, platform

i can use the following delay calculation to account for its clock bias.

$$\tilde{\phi}_i = \frac{1}{N_p} \sum_j \hat{\phi}_{i,j}. \quad (2.49)$$

Using the estimated clock bias, platform j corrects its transmit signal by applying the fractional delay $\tilde{\phi}_j$ along with a phase correction of $\exp(-j2\pi f^c \tilde{\phi}_j)$. Then, on reception at platform i , the signal is corrected by applying a fractional delay of $-\tilde{\phi}_i$ and a phase correction of $\exp(j2\pi f^c \tilde{\phi}_i)$. Together, these operations achieve time synchronization across the distributed radar network.

For carrier phase compensation, the estimates for all pairs (i, j) with $i \neq j$ can be calculated by

$$\hat{\gamma}_{i,j}^{err} = \angle d_{i,j}(m_{i,j}) + 2\pi f^c m_{i,j}. \quad (2.50)$$

Then using the carrier phase error estimates, the transmit carrier phase γ^{tx} and receive carrier phase γ^{rx} can be solved in a $N_p^2 - N_p + 1$ system where one equation is $\gamma_1^{tx} = 0$ and the rest of the linear equations are described by (2.5). Using the carrier phase estimates, each platform i compensates for carrier phase offsets by applying a phase shift of $-\gamma_i^{tx}$ to all its transmitted waveforms. Similarly, a phase shift of γ_j^{rx} is applied to all received signals at platform j ,

2.2.6 Fully Synchronized Signal Model

Once clock drift, clock bias, and carrier phase corrections are applied, the resulting distributed radar system is synchronized. The fully compensated baseband transmit waveform

is given by

$$\tilde{s}'(\tau_j) = s_j \left(\frac{\tau_j - \tilde{\phi}_i}{\hat{\alpha}_j} \right) \exp \left(j2\pi f^c \left(\frac{1}{\hat{\alpha}_j} - 1 \right) \tau_j \right) \exp \left(-j2\pi f^c \frac{\tilde{\phi}_j}{\hat{\alpha}_j} \right) \exp(-j\hat{\gamma}_j^{\text{tx}}). \quad (2.51)$$

The fully correct received signal on platform i yields

$$\begin{aligned} \tilde{r}_{i,j}(\tau'_i) = & s_j \left(\frac{\alpha_j \hat{\alpha}_i}{\hat{\alpha}_j \alpha_i} \tau'_i - \left(\frac{\alpha_j}{\hat{\alpha}_j \alpha_i} (\phi_i - \hat{\phi}_i) - \frac{1}{\hat{\alpha}_j} (\phi_j - \hat{\phi}_j) + \frac{\alpha_j}{\hat{\alpha}_j} \frac{R_{i,j}}{c} \right) \right) \\ & \cdot \exp \left(-j2\pi f^c \left(\frac{\alpha_j}{\hat{\alpha}_j \alpha_i} (\phi_i - \hat{\phi}_i) - \frac{1}{\hat{\alpha}_j} (\phi_j - \hat{\phi}_j) + \frac{\alpha_j}{\hat{\alpha}_j} \frac{R_{i,j}}{c} \right) \right) \\ & \exp \left(j2\pi f^c \left(\frac{\alpha_j \hat{\alpha}_i}{\hat{\alpha}_j \alpha_i} \tau'_i \right) \right) \exp(j((\gamma_j^{\text{tx}} - \hat{\gamma}_j^{\text{tx}}) - (\gamma_i^{\text{rx}} - \hat{\gamma}_i^{\text{rx}}))). \end{aligned} \quad (2.52)$$

Assuming ideal estimation of clock drift (i.e, $\alpha_i/\hat{\alpha}_i = \alpha_j/\hat{\alpha}_j$) and ideal estimation of clock bias and carrier phase (i.e, $\hat{\phi}_i = \phi$, $\hat{\gamma}_j^{\text{tx}} = \gamma_j^{\text{tx}}$, and $\hat{\gamma}_j^{\text{rx}} = \gamma_j^{\text{rx}}$), then (2.52) can be further reduced to

$$\tilde{r}_{i,j}(\tau'_i) = s_j \left(\tau'_i - \frac{R_{i,j}}{c} \right) \exp \left(-j2\pi f^c \frac{R_{i,j}}{c} \right). \quad (2.53)$$

This represents the ideal received signal after perfect compensation of clock drift, clock bias, and carrier phase.

2.3 Software Defined Radio

Realizing this signal model in practice requires flexible hardware capable of performing advanced signal processing. Software-defined radios (SDRs) offer such flexibility, as well as being cost-effective for implementing distributed radar synchronization. The main advantage of SDRs is their ability to turn issues that arise in hardware into the software domain, where they can be addressed through programming the radios. To enable this flexibil-

ity, most of the signal processing is performed through Field-Programmable Gate Arrays (FPGAs), which provide the computational speed needed for real-time operation. As a result, many algorithms and protocols can be implemented in SDRs in a manner comparable to traditional hardware radios. A wide range of SDR platforms are available, including the HackRF One, ADALM-PLUTO, and the Ettus Universal Software Radio Peripheral (USRP). For this project, the Ettus USRP was chosen due to its combination of flexibility and high performance. A detailed overview of the USRP and the hardware features relevant to this work is provided in the following subsection.

2.3.1 Universal Software Radio Peripheral

The USRP is commonly used in conjunction with the GNU Radio software framework, which will be discussed in Section 2.4. Developed and produced by Ettus Research and National Instruments, the USRP is made up of two main subdevices: the motherboard and one or more interchangeable daughterboards [23]. Depending on the motherboard, the daughterboards can be swapped to determine the available bandwidth and frequency coverage (ranging from 10 MHz to 8.4 GHz for the X-Series devices). For cost-efficiency and reliability, the motherboard chosen for this project was the Ettus X310 series.

The general architecture of the USRP is depicted in Figure 2.4. The system is organized into two main paths: transmit and receive. On the transmit side, the baseband signal is first generated in software before being passed through digital up-conversion (DUC) to convert the signal to an intermediate frequency (IF). The signal is then converted into analog using the digital-to-analog converters (DACs), where it can be filtered using a low-pass filter. Lastly, the signal is mixed up to radio frequency (RF) using the local oscillator (LO) via a phased-locked loop (PLL) and a voltage-controlled oscillator (VCO), then amplified and transmitted. On the other side, the receive chain passes the incoming RF signals through

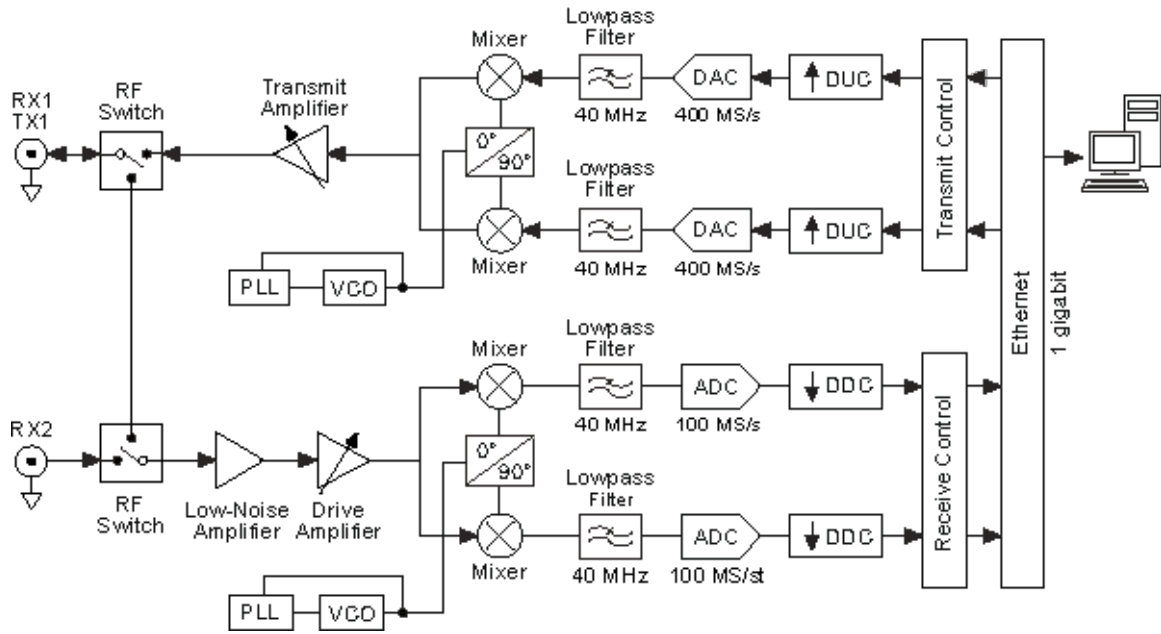


Figure 2.4: A general block diagram of USRP architecture. From [23].

low-noise and drive amplifiers, where it is then mixed down to IF. The signals are then filtered, digitized through the analog-to-digital converters (ADCs), and processed to base-band using the digital down-converters (DDCs). The signal is then received and passed to the host computer over a gigabit Ethernet interface. This architecture provides users with control of signal generation and processing, enabling them to cover a wide variety of communication and radar applications.

The UBX 160 daughterboard was chosen for this project due to its wide frequency coverage and large instantaneous bandwidth. It offers a frequency coverage from 10 MHz to 6 GHz with up to 160 MHz of instantaneous bandwidth. Figure 2.5 presents the block diagram of the UBX 160 daughterboard. On the transmit path, the baseband signal is upconverted to RF using either a direct-conversion path (500 MHz to 6 GHz) or a down-conversion path (10 MHz to 500 MHz). This allows for a wide range of frequency capabilities for the user. Similarly, the receive side takes in incoming RF signals, which are then amplified and filtered based on the path chosen. The signals are then downconverted using

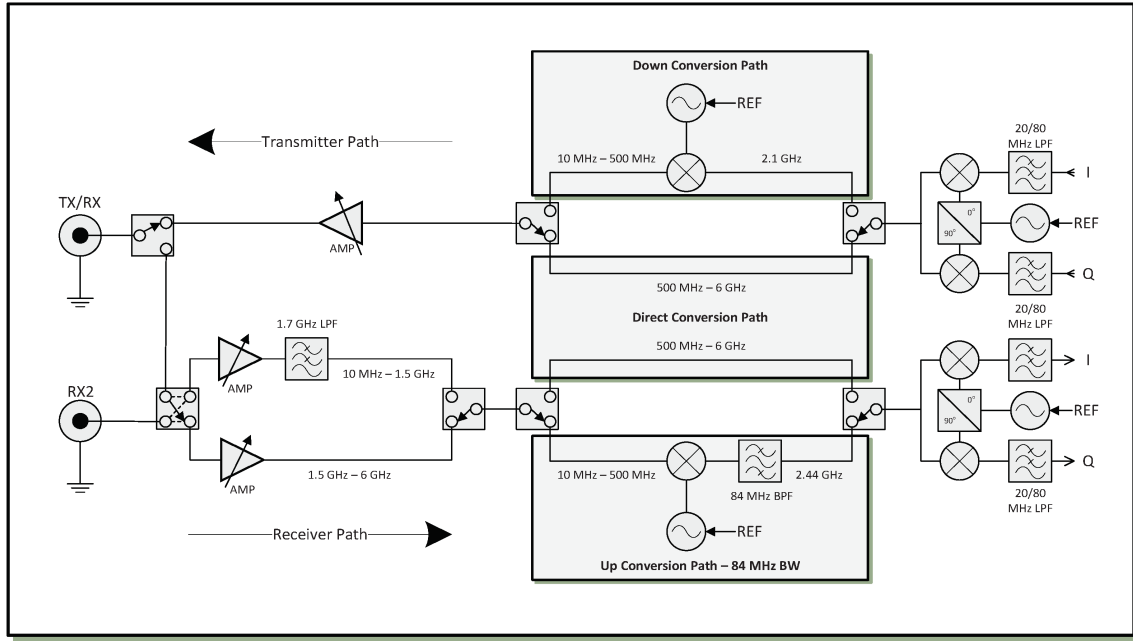


Figure 2.5: Block diagram of the UBX 160 daughterboard architecture. From [24].

either the direct-conversion path or the up-conversion path.

The USRP motherboards and daughterboards provide the necessary RF front-end and signal conversion capabilities; however, practical operations also require a software framework to configure the radios, implement signal processing, and interface with the host computer. The following section introduces GNU Radio, the software framework used in this project to implement the synchronization algorithms.

2.4 GNU Radio

GNU Radio is a free and open-source software platform for building signal processing systems on software-defined radios [25]. This makes GNU Radio a valuable tool for research, academia, industry, and hobbyists, offering a large collection of pre-built signal processing blocks. As such, a variety of pre-built functional blocks are listed below:

- Boolean operators including And, Not, Or, and Xor

- File operators such as File Meta Sink, File Meta Source, File Sink, File Source, Wav File Sink, and Wav File Source
- Filters including the Band Pass Filter, Band Reject Filter, DC Blocker, FFT Filter, Filter Delay, High Pass Filter, IIR Filter, and Low Pass Filter
- Fourier analysis blocks
- Instrumentation blocks including QT GUI Frequency Sink, QT GUI Sink, and QT GUI Time Sink
- Message Tools such as Message Debug, Message Strobe, and Variable to Message
- Modulators such as FM Demod, Frequency Mod, GFSK Demod, GFSK Mod, Phase Mod, and Quadrature Demod
- Protocol Data Unit (PDU) tools
- Type Converters including Complex to Float, Complex to Mag², Float to Complex, Float to Int, and Int to Float
- UHD blocks such as UHD: USRP Sink and UHD: USRP Source

Additionally, as an open-source platform, GNU Radio makes it possible for users to develop their own blocks through custom out-of-tree (OOT) modules for specific applications [25]. These modules can then be shared within the community, enabling collaboration in the communication and radar communities. The development of pre-built and OOT modules allows users to interweave different signal processing blocks together for their needs. The modular nature of the software is rooted in its architecture and will also influence the design of this project.

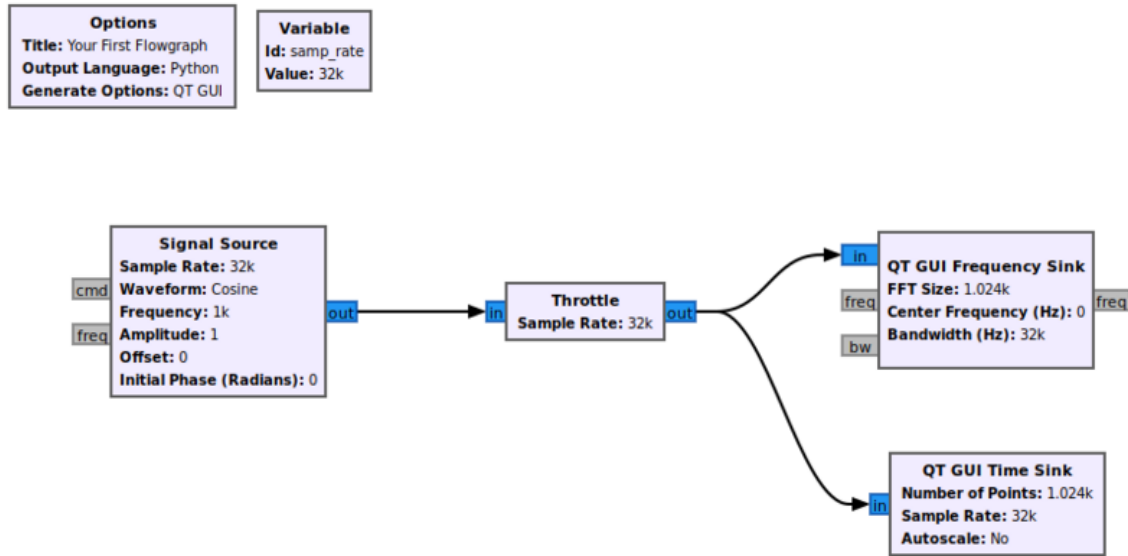


Figure 2.6: Example flowgraph in GNU Radio Companion. From [25].

2.4.1 GNU Radio Architecture

The GNU Radio architecture is primarily implemented through two programming languages, C++ and Python. The digital signal processing blocks are implemented in C++ for efficiency, while Python provides the framework for connecting these blocks, controlling data flow, and managing the runtime environment. Using both these languages allows blocks to be connected, creating complex signal processing applications that are constructed through modular components. Since the USRP hardware is designed to handle general RF front-end functions, such as up-conversions and down-conversions, GNU Radio will perform the majority of the digital signal processing through its blocks.

GNU Radio provides an Application Programming Interface (API) for controlling and configuring USRP devices. Its graphical tool, GNU Radio Companion (GRC), enables users to construct flowgraphs visually rather than through code alone. A flowgraph is a directed acyclic graph consisting of signal processing blocks and arrows that correspond to the data streams between them. This means that data between blocks flows in one direction

and does not form any cycles. An example of a flowgraph is shown in Figure 2.6. A flowgraph consists of interconnected signal processing blocks, where each block contains a defined number of inputs and/or outputs depending on its functionality. The input ports are where the block receives data, and once the data is processed internally, it exits through its output ports.

Typically, GNU Radio blocks operate in the continuous or stream domain, in which data is passed sequentially through buffers at a fixed rate. Alternatively, blocks can communicate using the message-passing domain, which supports both asynchronous and synchronous data exchange. Subsection 2.4.2 will discuss the inner workings of the messaging domain and how it is applied to radar applications.

While the GNU Radio framework lays out the data exchange and the signal processing, it must also communicate with the external hardware. That is the role of the USRP Hardware Driver (UHD), which serves as the low-level communication layer between the GNU Radio flowgraphs and the USRP devices. Figure 2.7 illustrates the relationship between GNU Radio, UHD, and the USRP. UHD allows for the configuration of the radios' parameters, such as sampling rate, center frequency, and gain, and also manages the streaming of samples to and from the SDRs. In other words, UHD coordinates the transmit and receive operations, enabling integration of the digital signal processing with the RF front-end of the radios.

2.4.2 GNU Radio Implementation of Pulsed Radar via the Message Domain

The development of GNU Radio modules for pulsed radar is a rapidly expanding area for researchers and industry, as the demand for software-defined radio radar systems rises. An example pulsed radar toolkit that also served as the basis for this research is the *gr-plasma*

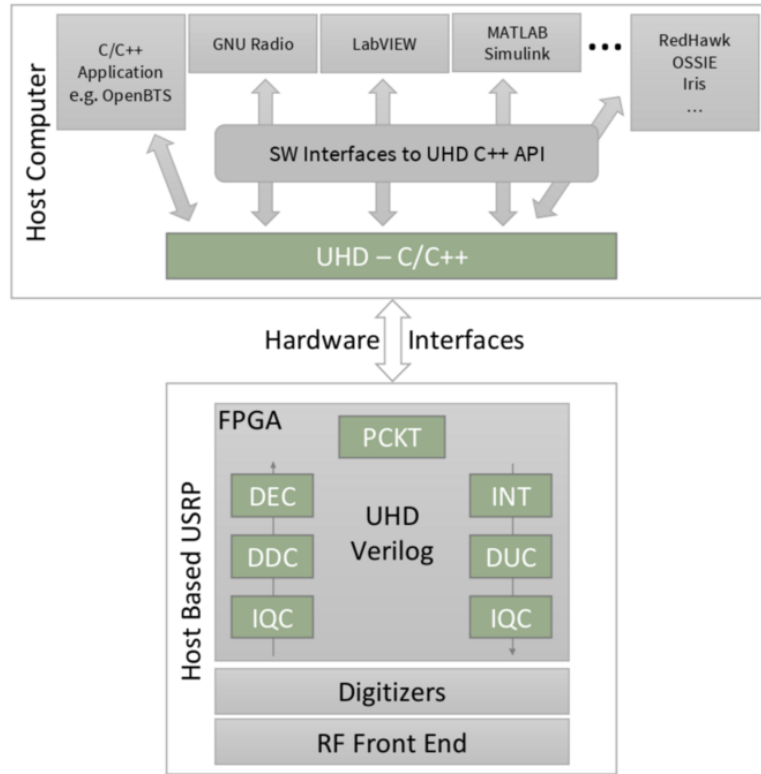


Figure 2.7: Block diagram of UHD and its role between GNU Radio and USRP hardware. From [23].

OOT module [26]. The module provides blocks necessary for radar signal processing functions, such as CFAR detection and Doppler processing, that are readily available to the public. *gr-plasma* functions entirely in the message domain, which is needed for pulsed radar applications. The operations of pulsed radars require that transmitted signals be turned on and off with precise timing, enabling detection and measurements of targets. As such, the streaming domain will not be used in this research as it does not provide the essential tools to generate pulsed signals.

The message domain communicates through Polymorphic Types (PMTs), which are data types that act like generic containers of data that can be passed between blocks [25]. PMTs can represent a variety of data types, including:

- Boolean values - logical true or false.

- Strings - symbolic text or identifiers.
- Integers - signed or unsigned whole numbers.
- Floats - real-valued samples.
- Complex - complex-valued samples.
- Pairs - two PMTs combined.
- Vectors - ordered list of PMT values.
- Dictionaries - key-value mappings for metadata.

These data types serve as the foundation of the message domain, supporting both simple control messages and structured data packets.

A special form of PMTs that is widely used in the message domain is the protocol data unit (PDU) [25]. It is a pair consisting of a PMT dictionary and a PMT vector. The dictionary portion of the PDU provides the contextual information, known as metadata, that describes the accompanying data vector. This will be important for differentiating estimates for clock drift, clock bias, and carrier phase, as well as identifying the current stage of the synchronization process in this project. Furthermore, the PMT vector will be essential for carrying the signal content of the transmitted and received waveforms. Because this project operates entirely in the message domain, all waveform transmission and reception are handled through discrete data packets, or in this case, PDUs.

Each PDU corresponds to a single event, whether it be a transmitted or received signal. These events are handled using burst sample mode, a feature in UHD that enables precisely timed, packet-based communication [25]. Burst mode allows the Ettus USRPs to transmit short segments of data, known as bursts, at precise times using metadata. This ensures control over when each transmission and reception begins and ends, allowing for the

scheduling of radar pulses. By using UHD's burst mode, the transmit and receive chains of a system can be coordinated within nanoseconds of accuracy. Unfortunately, all timing commands are quantized to the tick rate of the device's master clock. For example, a USRP operating with a 100 MHz master clock has a timing resolution of 10 nanoseconds per tick. To refine the timing beyond this quantization, an FFT-based fractional delay technique is used to interpolate the transmit and receive signals, achieving sub-nanosecond precision.

2.4.3 Fast Fourier Transform-Based Fractional Delay

In digital signal processing, a fractional delay of a signal refers to a shift in time by a non-integer number of samples [27]. This tool is essential for radar applications such as beamforming and synchronization, where precise sub-sample adjustments are required. One method of fractionally delaying a waveform is through the frequency-domain method known as the Fast Fourier Transform (FFT)-based fractional delay. This method exploits the time-shifting property of the Fourier transform to shift a signal in time.

Consider a discrete-time signal $x[n]$ with its frequency domain representation $X(\omega)$. A time delay of k samples can be represented by $x(n - k)$. Applying a discrete-time Fourier transform (DTFT) yields

$$\mathcal{F}[x(n - k)] = \sum_{n=-\infty}^{\infty} x(n - k)e^{-j\omega n} = e^{-j\omega k} X(\omega). \quad (2.54)$$

The FFT-based fractional delay technique leverages the Fourier transform property, which states that a delay in time corresponds to a linear phase shift in frequency [27]. The signal is transformed into the frequency domain, phase-shifted by $e^{-j\omega k}$ (where k may be fractional), and converted back using the inverse FFT to achieve a precise fractional time delay. This allows for sub-sample adjustments needed for synchronization later.

Chapter 3

Implementation of Synchronization Algorithm with USRP and GNU Radio

This chapter outlines the objectives and overall framework of the `gr-harmonia` OOT module. It begins by describing the objectives that influence the implementation of the synchronization algorithm. The following sections present the organization and functionality of each block developed for this module. The discussion of the blocks describes the parameters, data types, computations, and input/output structures created to enable synchronization across the radar systems. The structuring of the flowgraph using these blocks is then explained, followed by a discussion of the module's current limitations.

3.1 Objectives

The primary objective of the module is to demonstrate the synchronization algorithm's capability to achieve clock drift, clock bias, and carrier phase coherency among three software-defined radios (SDRs). To accomplish this, the module was designed to operate on a single host computer, where all SDR platforms remained stationary throughout testing. Using this configuration served as a controlled environment to validate the performance of the algorithm without introducing platform motion as a variable.

Another goal of the module is to make the system modular. Each part of the synchro-

nization process was built as its own GNU Radio block with clear inputs and outputs. This makes it easier to test and replace individual parts of the system without changing everything else. The modular design also makes the system easier to expand in the future, such as adding more SDRs or testing moving platform scenarios.

3.2 gr-harmonia: A Custom Out-of-Tree Module for SDR Synchronization

The design of `gr-harmonia` focuses on modularity and ease of use for research and experimental use. Each block was developed to operate within the message domain, enabling support for pulsed radar applications and simplifying the exchange of synchronization estimates [28]. Additionally, the blocks were structured to isolate specific stages of the synchronization process for analysis, debugging, and substitution with alternative estimation techniques. This will be helpful in the further development of the module, especially when platform motion is introduced.

The module was divided into eight distinct processing blocks: Single-Tone Source, LFM Source, UHD: All Radar, Frequency Peak Estimation, Clock Drift Estimation, Time Peak Estimation, Clock Bias and Carrier Phase Estimation, and Waveform Compensation. All synchronization estimates and intermediate signals are transferred as PDUs, ensuring consistent message-based communication and simplifying both data logging and computation [29]. The following subsections describe the internal functionality of each block.

3.2.1 Waveform Generation

The synchronization process begins with waveform generation, where the signals are created for transmission between platforms to estimate clock drift, clock bias, and carrier phase offsets. Within the `gr-harmonia` module, this functionality is implemented by

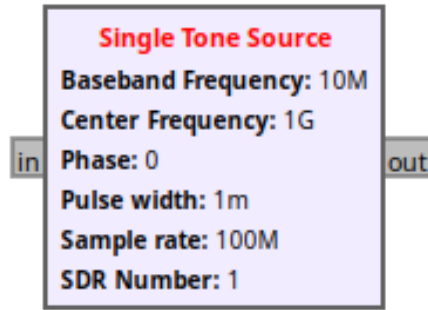


Figure 3.1: Single-Tone Source Block. The block generates a single-tone frequency waveform at baseband.

the Single-Tone Source and LFM Source blocks. These blocks form the foundation for all the following synchronization stages by providing baseband signals for estimation and analysis.

The Single-Tone Source block generates a complex baseband sinusoidal waveform used for frequency estimation. The single-frequency tone, defined by (2.13), is formed within the block and transmitted through its output port as a PDU. The block includes several configurable parameters: baseband frequency, center frequency, phase, pulse width, sample rate, and SDR number, as shown in Figure 3.1. These parameters are used to adjust the waveform while embedding the transmitting SDR ID number as metadata for waveform association within the UHD block. Furthermore, the block labels the waveform as a "single-tone" for identification in downstream processing. The list of metadata keys and their respective descriptions is shown in Table 3.1.

An example of the Single-Tone Source block outputting a 10 kHz waveform, plotted using GNU Radio's QT GUI Time Sink block, is shown in Figure 3.2. This block plots the waveform into its real and imaginary parts, illustrated in blue and red, respectively. For testing, a 10 MHz signal was set for all single-tone waveform generation.

The LFM Source block produces a linearly frequency-modulated waveform used for

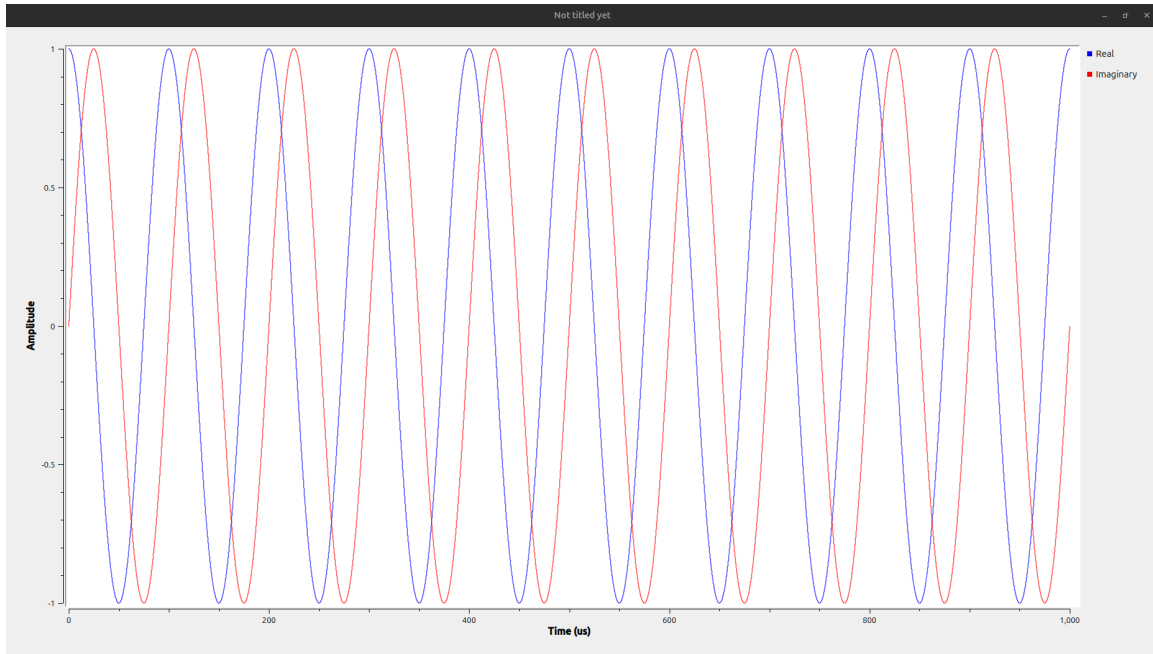


Figure 3.2: Example output of a single-tone block.

Table 3.1: Possible Metadata Keys for an Output PDU of the Single-Tone Block.

Metadata Key	Description
PMT_HARMONIA_LABEL	Defines the waveform with the value "single_tone" to identify the waveform type.
PMT_HARMONIA_SDR1	Generated only if the SDR number equals 1; otherwise, this key is not included in the metadata.
PMT_HARMONIA_SDR2	Generated only if the SDR number equals 2; otherwise, this key is not included in the metadata.
PMT_HARMONIA_SDR3	Generated only if the SDR number equals 3; otherwise, this key is not included in the metadata.

clock bias and carrier phase estimation. The signal is defined by (2.41), and its generation is controlled by the following parameters: bandwidth, starting frequency, center frequency, pulse width, second pulse width, sample rate, and SDR number, as illustrated in Figure 3.3. These parameters enable users to adjust the design of the LFM waveform while embedding the associated transmitting SDR ID number and waveform type as metadata for later use in the UHD block. An example output of an LFM block is shown in Figure 3.4.

To accommodate different estimation stages, the block includes two pulse width parameters, allowing the waveform duration to be adjusted independently for clock bias and carrier phase estimation. Additionally, the block features three input ports: clock drift, clock bias, and carrier phase, shown on the left side of the block. These inputs enable compensation for corresponding synchronization estimates in the transmitted signal when available. The signal is then output as a PDU containing metadata shown in Table 3.2. The metadata dictionary of the LFM Source block contains several keys that describe waveform characteristics and synchronization parameters. The key `PMT_HARMONIA_LABEL` defines the waveform type by assigning the string value "lfm", allowing the waveform to be identified in the UHD block for signal organization. The keys `PMT_HARMONIA_SDR1`, `PMT_HARMONIA_SDR2`, and `PMT_HARMONIA_SDR3` are generated on the condition that the user-defined SDR number parameter matches its key name. Additionally, when clock drift estimates are present, these keys will hold the corresponding estimate for that SDR. Similarly, `PMT_HARMONIA_CB_SDR1`, `PMT_HARMONIA_CB_SDR2`, and `PMT_HARMONIA_CB_SDR3` store the clock bias estimates for their respective SDRs when applicable. The same is said for the keys `PMT_HARMONIA_CP_TX_SDR1`, `PMT_HARMONIA_CP_TX_SDR2`, and `PMT_HARMONIA_CP_TX_SDR3`, which store the transmit carrier phase estimates. Lastly, the Boolean flags `clock_drift_enable`, `clock_bias_enable`, and `carrier_phase_enable` indicate the estimates applied to the waveform for downstream processing.

Together, these waveform generation blocks provide the basis for testing and validating the synchronization algorithm. Both generate their respective signal when the flowgraphs begin; however, the LFM block updates its waveform whenever a new estimate arrives. Every signal is formatted as a PDU containing the metadata and signal vector, ensuring that each waveform is correctly routed to the subsequent synchronization stage within the UHD block.

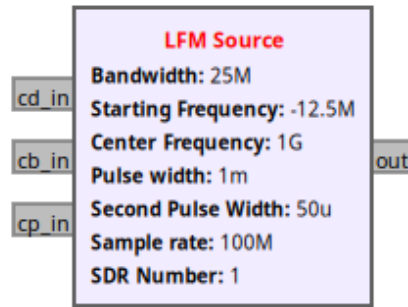


Figure 3.3: LFM Source Block. Generates a linearly frequency-modulated waveform at baseband.

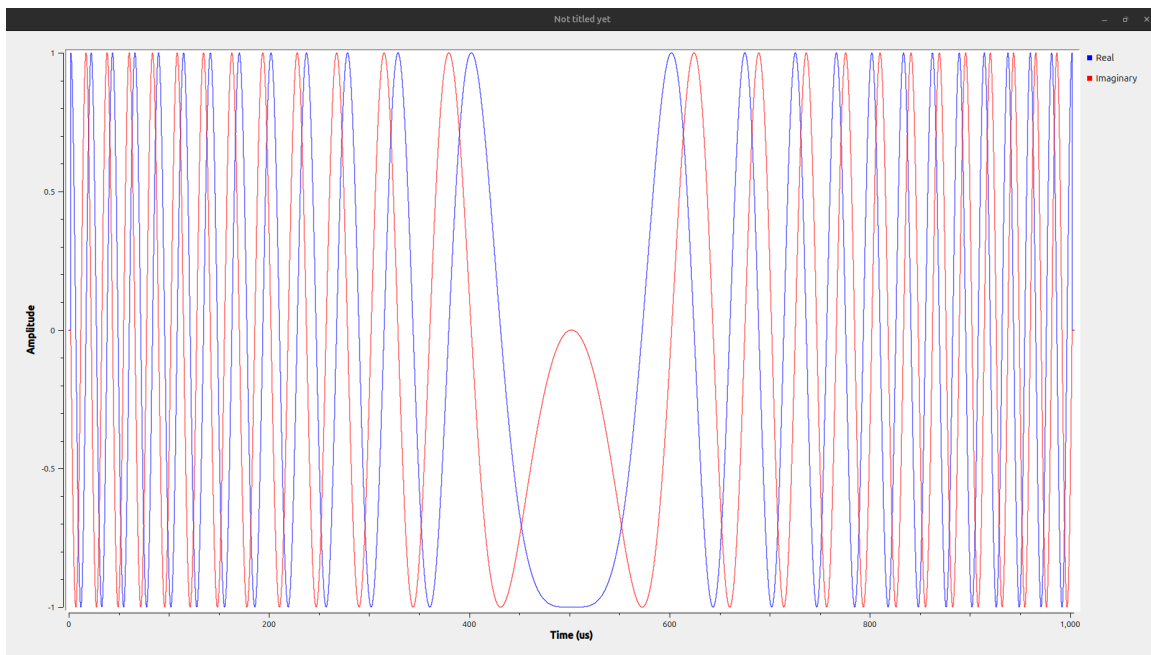


Figure 3.4: Example waveform output of an LFM block.

3.2.2 USRP Interfacing and Control

The UHD: All USRP Radars block serves as the hardware interface between the gr-harmonia module and the physical USRP devices. This block manages both the waveform transmission and reception while maintaining timing control of each SDR platform. The block uses the UHD API to schedule timed bursts with nanosecond-level pre-

Table 3.2: Possible Metadata Keys for an Output PDU of the LFM Block.

Metadata Key	Description
PMT_HARMONIA_LABEL	Defines the waveform with the value "lfm" to identify the waveform type.
PMT_HARMONIA_SDR1	Generates the key and attaches the clock drift estimate for SDR 1 only if the SDR number equals 1; otherwise, this key is not included in the metadata.
PMT_HARMONIA_SDR2	Generates the key and attaches the clock drift estimate for SDR 2 only if the SDR number equals 2; otherwise, this key is not included in the metadata.
PMT_HARMONIA_SDR3	Generates the key and attaches the clock drift estimate for SDR 3 only if the SDR number equals 3; otherwise, this key is not included in the metadata.
PMT_HARMONIA_CB_SDR1	Generates the key and attaches the clock bias estimate for SDR 1 only if the SDR number equals 1; otherwise, this key is not included in the metadata.
PMT_HARMONIA_CB_SDR2	Generates the key and attaches the clock bias estimate for SDR 2 only if the SDR number equals 2; otherwise, this key is not included in the metadata.
PMT_HARMONIA_CB_SDR3	Generates the key and attaches the clock bias estimate for SDR 3 only if the SDR number equals 3; otherwise, this key is not included in the metadata.
PMT_HARMONIA_CP_TX_SDR1	Generates the key and attaches the transmit carrier phase estimate for SDR 1 only if the SDR number equals 1; otherwise, this key is not included in the metadata.
PMT_HARMONIA_CP_TX_SDR2	Generates the key and attaches the transmit carrier phase estimate for SDR 2 only if the SDR number equals 2; otherwise, this key is not included in the metadata.
PMT_HARMONIA_CP_TX_SDR3	Generates the key and attaches the transmit carrier phase estimate for SDR 3 only if the SDR number equals 3; otherwise, this key is not included in the metadata.
clock_drift_enable	Outputs true if the block has received clock drift estimates, otherwise outputs false.
clock_bias_enable	Outputs true if the block has received clock bias estimates, otherwise outputs false.
carrier_phase_enable	Outputs true if the block has received carrier phase estimates, otherwise outputs false.

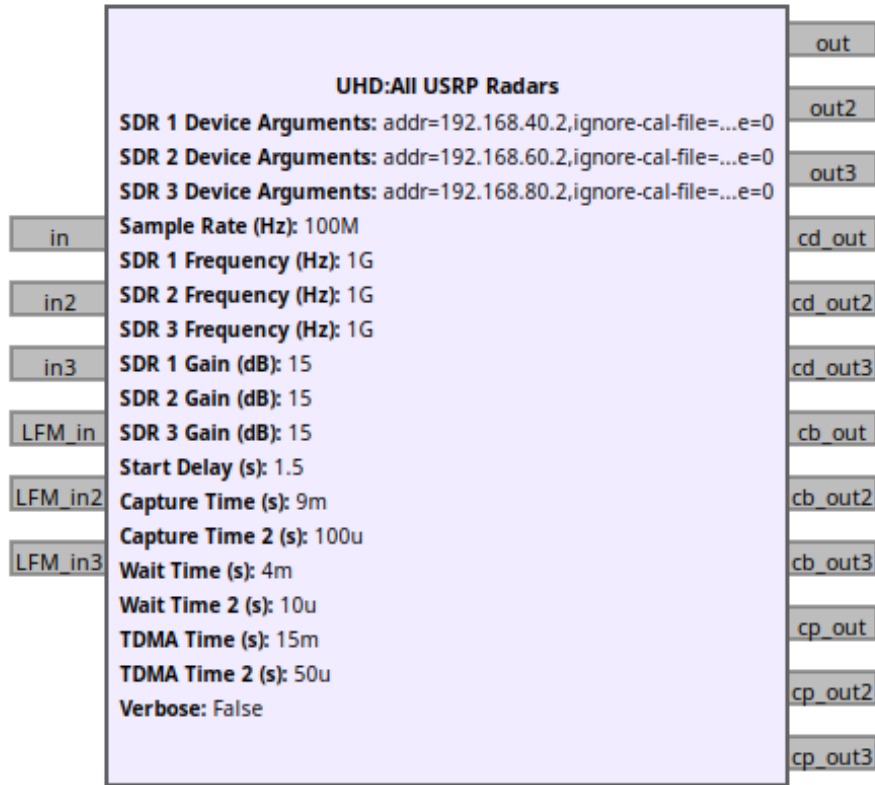


Figure 3.5: UHD: All USRP Radars Block. Controls all signal exchanges between a three-SDR network and the configuration of each radar.

cision, enabling accurate timing of radar operations and supporting TDMA-based signal exchanges between platforms.

The block provides several parameters that define the timing and hardware configuration for a three-SDR system, as shown in Figure 3.5. These parameters are: SDR device arguments, sample rate, SDR frequency, SDR gain, start delay, capture time/capture time 2, wait time/wait time 2, TDMA time/TDMA time 2, and verbose. Given these parameters, they enable precise control over USRP configurations, transmission timing, reception windows, and debugging control.

Starting with the USRP configurable parameters, the SDR device arguments parame-

ter specifies the device address (IP) of each USRP in the system. Each SDR is configured independently through its corresponding IP address (e.g., 192.168.40.2, 192.168.60.2, 192.168.80.2), allowing the block to communicate with specific USRP devices. The sample rate parameter defines the sampling frequency (in Hz) used for transmission and reception. SDR frequency sets the RF center frequency for each SDR, determining the operating frequency of the transmit and receive chains. Then, SDR gain controls the analog front-end gain for both transmission and reception to ensure proper signal power levels.

The following parameters are used to configure the timing of the transmit and receive chains for each SDR. To begin, the start delay parameter was introduced as an initial timing offset of SDR operations due to hardware initialization and buffering differences among the SDRs. This ensures all nodes are ready to initiate the synchronization process. The capture time parameter defines the receive window during the clock drift and clock bias estimation stages. It is used to determine the length of time each SDR is capturing data. Similarly, the capture time 2 is used for the carrier phase estimation stage. Due to clock bias offsets, wait time/wait time 2 was created as an intermediate waiting period to forward the receive time of each SDR, to ensure each SDR captures the transmitted signals. Following up, the TDMA time/TDMA time 2 parameters control the transmit time spacing between SDRs during the signal-exchange stages. These parameters follow (2.9) and (2.11) to ensure that only one SDR transmits at a time, enabling orthogonal TDMA operation. Finally, verbose is used to create detailed console outputs of configuration parameters and timing information for debugging.

The functionality of the block begins at startup when the SDRs are initialized with their addresses, center frequency, gain, and the resetting of their internal clocks. Using UHD's `set_time_now()` command, the internal clocks of all the SDRs are reset to zero. However, due to processing latency, each device experiences a small delay of a couple of milliseconds with respect to the other. The synchronization signal exchange begins

shortly afterwards when a PDU is received by the block, representing a single transmit event. The metadata portion of the PDU contains contextual information such as waveform identifiers, SDR numbers, synchronization estimates (if available), and control flags (e.g., `clock_drift_enable`, `clock_bias_enable`, and `carrier_phase_enable`). This information will ensure the block is operating at the correct synchronization stage. Additionally, the data vector portion of the PDU holds the complex baseband samples that are used for transmission.

During transmission, the block interprets the metadata field of the PDU to form a UHD transmit command. If `clock_drift_enable` is true, the block is set to the clock drift stage, which adjusts the burst timing according to the clock drift estimate and the SDR it is supposed to be applied to. Similarly, if `clock_bias_enable` or `carrier_phase_enable` are present, the block is set to the clock bias or carrier phase stage, and then the corresponding offset corrections are applied to the scheduled transmit time. Additionally, the transmit signals undergo an FFT-based fractional delay correction, defined by (2.54), based on the quantization of UHD and timing delay estimates. Each burst is then transmitted using a UHD time command based on the configured timing parameters. It is important to note that, because the waveforms are transmitted OTA, the physical waveform cannot carry any metadata information. During reception, the block operates in burst mode to capture data within a specific time window determined by the synchronization stage and parameter configurations. Once the signal is received, it is combined with the metadata of the calculated receive time error shown in Table 3.3. The received signal and metadata are then packaged into PDUs and passed downstream to the Waveform Compensation block, which applies any necessary compensations to the received signals.

As illustrated in Figure 3.5, the UHD block features dedicated message ports for the input and output PDUs for each SDR. The input port accepts waveforms generated by the Single-Tone Source or LFM Source blocks, while the output ports produce the received

Table 3.3: Metadata Key for an Output PDU of the UHD Block.

Metadata Key	Description
<code>rx_error</code>	Contains the residual timing error due to quantization and synchronization compensation, used for subsequent processing in the waveform compensation block.

signal PDUs corresponding to the receiving SDR and current estimation stage. This design allows each synchronization stage to be output separately, making debugging and signal analysis easier across the processing chain.

3.2.3 Waveform Compensation

The Waveform Compensation block implements the compensation stage of the synchronization algorithm within `gr-harmonia`, by applying the estimated synchronization parameters to the received signals. This block adjusts the received signals of each SDR by compensating for clock drift, clock bias, and carrier phase errors when their estimates are calculated. Seen in Figure 3.6, the block receives four primary inputs in the form of PDUs: the received signal, clock drift estimates, clock bias estimates, and carrier phase estimates. Each of these PDUs contains metadata identifying the correction parameters and the corresponding SDR they should be applied to. As mentioned in the previous subsection, the received signal PDU contains the `rx_error` metadata field, which stores the residual timing error. This metadata value provides the timing offset needed for timing compensation in this block.

When the block receives a received signal PDU through the "in" port, it converts the complex baseband waveform into the frequency domain using an FFT. This allows the block to apply the FFT-based fractional delay to the signal to compensate for the timing offset on the receive end of the signal exchange. After the signal is converted back into the time domain, the block applies the phase offsets caused by clock drift, clock bias, and

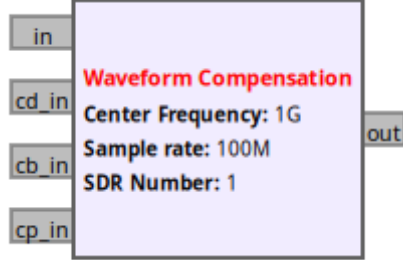


Figure 3.6: Waveform Compensation Block. The block adjusts received waveforms with clock drift, clock bias, and carrier phase estimates.

carrier phase estimates to the signal. The resulting waveform is then packaged into a new PDU, along with its original metadata, for further processing in downstream blocks.

3.2.4 Clock Drift Estimation Blocks

The Frequency Peak Estimation and Clock Drift Estimation blocks form the next stage of the synchronization process within `gr-harmonia`. These two blocks operate sequentially, with the Frequency Peak Estimation block executed first. Their primary purpose is to determine the frequency of the received single-tone waveforms and compute the resulting clock drift of each platform relative to a reference SDR (e.g., SDR 1).

The Frequency Peak Estimation block, shown in Figure 3.7, includes the following parameters: FFT ratio, pulse width, capture length, sample rate, NL-LS iterations, and platform number. These parameters enable the refinement of the frequency estimation process through the sinc-based NL-LS estimation technique and the formation of a frequency array that correlates with the received signal. The platform number parameter specifies which SDR is performing the estimation and is stored under the metadata key `"rx_id"`.

The block begins operations when it receives PDUs containing the received single-tone waveforms from the UHD: All Radar block. Each PDU includes metadata identifying the transmitting SDR, allowing frequency estimates to be associated with the cor-

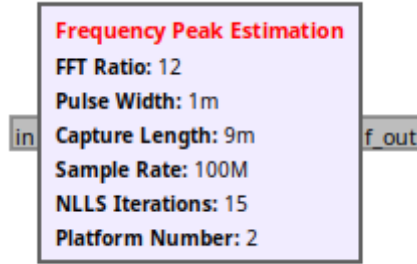


Figure 3.7: Frequency Peak Estimation Block. The block estimates the frequency of single-tone signals and assigns them to the corresponding receiving SDR.

Table 3.4: Metadata Keys for an Output PDU of the Frequency Peak Estimation Block.

Metadata Key	Description
rx_id	Defines the PDU with the corresponding receiving SDR ID.
PMT_HARMONIA_SDR1	Holds the frequency estimate from SDR 1's transmitted signal.
PMT_HARMONIA_SDR2	Holds the frequency estimate from SDR 2's transmitted signal.
PMT_HARMONIA_SDR3	Holds the frequency estimate from SDR 3's transmitted signal.

responding metadata key seen in Table 3.4. The block performs an FFT on the received signal to obtain its frequency spectrum and identifies the frequency bin corresponding to the dominant peak. This peak frequency is further refined using the sinc-based NL-LS estimator, which iterates several times based on the NL-LS iteration parameters. The measured frequency is then stored in the metadata dictionary under PMT_HARMONIA_SDR1, PMT_HARMONIA_SDR2, and PMT_HARMONIA_SDR3, depending on which SDR transmitted the signal. The transmission order should remain consistent with the predetermined TDMA schedule known on each SDR.

The Clock Drift Estimation block follows immediately after the Frequency Peak Estimation block and uses the extracted frequencies to compute the clock drift values for each SDR. Since communication signals are not implemented in this project, frequency esti-

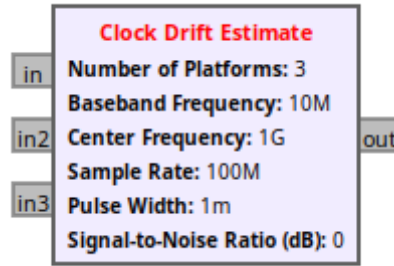


Figure 3.8: Clock Drift Estimation Block. Estimates the clock drift using the frequency estimates from the Frequency Peak Estimation Block.

mates are instead "exchanged" virtually within the GNU Radio Companion software. This is achieved by creating three Frequency Peak Estimation blocks, each simulating a separate SDR, that feed into their corresponding "in" ports on the Clock Drift Estimation block, as shown in Figure 3.8. The block contains the following parameters: number of platforms, baseband frequency, center frequency, sample rate, pulse width, and SNR ratio. The number of platforms, center frequency, and baseband frequency parameters are used to define the system of equations expressed in (2.19), which models the clock drift relationship between transmit and receive platforms. The remaining parameters support the clock drift process in the least-squares estimation. Although the SNR parameter is not required for the current implementation, it is retained for future development using the WLS approach.

In particular, the block computes the clock drift estimates using the OLS approach. Once computed, the estimates are inserted into the metadata dictionary under the keys `PMT_HARMONIA_SDR1`, `PMT_HARMONIA_SDR2`, and `PMT_HARMONIA_SDR3`, which are summarized in Table 3.5. These entries represent the calculated clock drift values for each SDR. The metadata entries are then sent in a PDU with an empty vector to be used later in the LFM Source, Waveform Compensation, and UHD blocks. A Boolean flag, `clock_drift_enable`, is then sent to notify these blocks to apply the given clock drift estimates and initiate the subsequent synchronization stage.

Table 3.5: Metadata Keys for an Output PDU of the Clock Drift Estimation Block.

Metadata Key	Description
PMT_HARMONIA_SDR1	Holds the clock drift estimate for SDR 1.
PMT_HARMONIA_SDR2	Holds the clock drift estimate for SDR 2.
PMT_HARMONIA_SDR3	Holds the clock drift estimate for SDR 3.
clock_drift_enable	Indicates to the LFM and UHD blocks to apply the estimated clock drift estimates for timing correction.

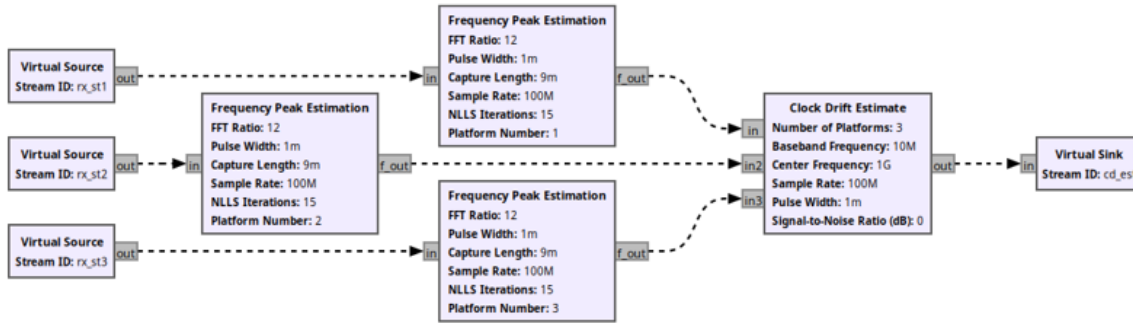


Figure 3.9: Flowgraph connection between the Frequency Peak Estimation blocks and the Clock Drift Estimation block.

Figure 3.9 illustrates how the Frequency Peak Estimation blocks interface with the Clock Drift Estimation block to simulate the exchange of the estimates within the synchronization flowgraph. The output connections of the Time Peak Estimation blocks are the refined time estimates that are used as the inputs of the Clock Drift Estimation block. By passing the estimates forward, the clock drift estimates are computed and sent through a virtual sink for compensation in further processes.

3.2.5 Clock Bias and Carrier Phase Estimation Blocks

Clock bias and carrier phase estimation are the last stage of the synchronization estimation process within `gr-harmonia`. The two blocks that incorporate this stage are the Time Peak Estimation block and the Clock Bias and Carrier Phase Estimation block. These blocks operate in the time domain and work sequentially to estimate the timing and phase

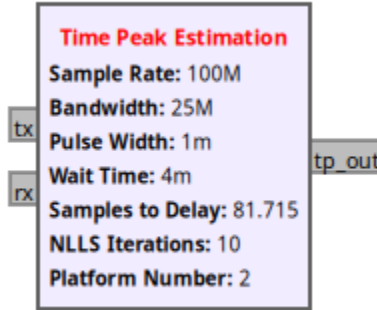


Figure 3.10: Time Peak Estimation Block. The block estimates time and phase based on the received pulse compression waveform.

offsets between the platforms after clock drift has been compensated for. Similar to the clock drift estimation process, the Time Peak and Clock Bias and Carrier Phase Estimation blocks also operate entirely within the message domain, but differ in that they estimate for time rather than frequency. The following subsection describes how `gr-harmonia` processes data to determine clock bias and carrier phase offsets.

The Time Peak Estimation block estimates the relative time-of-arrival (TOA) between a transmitted and received LFM signal using matched filtering. As shown in Figure 3.10, the block features two input ports: one for the transmitted signal ("tx") and one for the received signal ("rx"). These inputs are processed through a matched-filter operation once both waveforms are received. This allows the block to measure the timing delay between platforms to be exchanged on GNU Radio. The output of the block provides the time delay, $m_{i,j}$, and peak phase value, $\angle d_{i,j}(m_{i,j})$, for use in the Clock Bias and Carrier Phase Estimation Block.

The Time Peak Estimation block includes the following parameters: sample rate, bandwidth, pulse width, wait time, samples to delay, NL-LS iterations, and platform number. The sample rate defines the resolution of the time array used to estimate the arrival time of the received signal. In conjunction with the bandwidth, it is also used to define the ini-

tial condition of the NL-LS estimator. The pulse width, wait time, and samples to delay parameters are used to adjust the time array to account for the match filter, receive time window offsets, and front-end latency introduced by the USRP hardware. The NL-LS iterations parameter controls the number of iterations performed by the NL-LS refinement to further improve the accuracy of the time estimate. Finally, the platform number identifies the receiving SDR and is stored as metadata under the key "rx_id".

The metadata dictionary for the time block, seen in Table 3.6, contains several keys used to identify the receiving SDR and store the corresponding time and phase estimates to their transmitter. As mentioned before, the key `rx_id` defines the PDU with the identifier of the receiving SDR, allowing the Clock Bias and Carrier Phase estimation block to associate each estimate with the correct platform for future calculations. The keys `PMT_HARMONIA_SDR1`, `PMT_HARMONIA_SDR2`, and `PMT_HARMONIA_SDR3` store the time estimates obtained from the signal exchange of the LFM signals after NL-LS. Similarly, the keys `PMT_HARMONIA_P_SDR1`, `PMT_HARMONIA_P_SDR2`, and `PMT_HARMONIA_P_SDR3` store the corresponding phase estimates calculated at the matched-filter peaks. For every Time Peak Estimation block, there are two time estimates and two phase estimates available, which are used in the next block, Clock Bias and Carrier Phase Estimation.

The Clock Bias and Carrier Phase Estimation block is the final estimation stage in the synchronization chain within `gr-harmonia`. This block calculates the clock bias and carrier phase offsets after clock drift has been compensated and when all time and phase estimates are processed. Accurate estimation of these parameters ensures coherent signal integration across platforms once they have been compensated for. Shown in Figure 3.11, the Clock Bias and Carrier Phase Estimation block contains four inputs: three for the time and phase estimates obtained from each SDR, and one for the clock drift estimates. These inputs are combined to form the necessary connections between platform for synchroniza-

Table 3.6: Metadata Keys for an Output PDU of the Peak Time Estimation Block.

Metadata Key	Description
rx_id	Defines the PDU with the corresponding receiving SDR ID.
PMT_HARMONIA_SDR1	Holds the time estimate from SDR 1's transmitted signal.
PMT_HARMONIA_SDR2	Holds the time estimate from SDR 2's transmitted signal.
PMT_HARMONIA_SDR3	Holds the time estimate from SDR 3's transmitted signal.
PMT_HARMONIA_P_SDR1	Holds the phase estimate from SDR 1's transmitted signal.
PMT_HARMONIA_P_SDR2	Holds the phase estimate from SDR 2's transmitted signal.
PMT_HARMONIA_P_SDR3	Holds the phase estimate from SDR 3's transmitted signal.

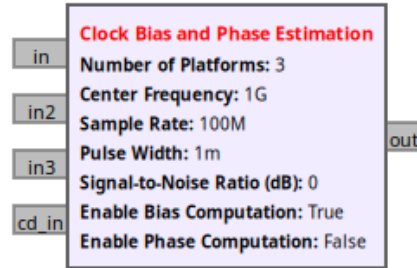


Figure 3.11: Clock Bias and Carrier Phase Estimation Block. Calculates the clock bias and carrier phase offsets based on the time and phase estimates from the Peak Time Estimation Block.

tion, enabling the block to compute the clock bias and carrier phase errors. The block outputs PDUs containing these estimates for each SDR, which are eventually used in the Waveform Compensation and UHD blocks for post synchronization purposes.

The block includes several configurable parameters such as the number of platforms, center frequency, sample rate, pulse width, and SNR Ratio. The number of platforms parameters is used to generate the matrix of delay values, $m_{i,j}$, enabling the block to compute the range and clock bias differences given by (2.47) and (2.48). Once these variables have

been computed, (2.49) is used to estimate the clock bias with respect to the network mean. The center frequency parameter is used for the estimation of phase, where the system of equations defined by (2.5) and (2.50) are applied to calculate the transmit and receive carrier phases. The remaining parameters are included for future implementations of WLS. While these parameters are not used in the current OLS approach, they provide flexibility for expanding the accuracy of the estimations in future work.

An option was provided to enable clock bias and carrier phase estimation independently. This feature allows the network to correct the clock bias before performing carrier phase estimation, ensuring that the SDRs are aligned in time within a fraction of the carrier wavelength and minimizing phase variation during the phase estimation process. This step is especially critical at higher operating frequencies, where small timing errors can introduce significant phase shifts if left uncorrected. In this project, the clock bias is estimated during one signal exchange, while the carrier phase is estimated during a subsequent exchange of LFM's to reduce phase drift and improve the estimation.

Once the block computes the final estimates of clock bias and/or carrier phase for all three SDRs, the values are stored in the metadata dictionary for downstream processing in the Waveform Compensation, LFM, and UHD blocks. First, the keys `PMT_HARMONIA_CB_SDR1`, `PMT_HARMONIA_CB_SDR2`, and `PMT_HARMONIA_CB_SDR3` hold the clock bias estimates for SDR 1, SDR 2, and SDR 3, respectively. Then, the keys `PMT_HARMONIA_R_SDR12`, `PMT_HARMONIA_R_SDR13`, and `PMT_HARMONIA_R_SDR23` store the range estimates between the SDR pairs, which are used to fully compensate the waveforms for synchronization. The transmit carrier phase estimates are stored under the keys `PMT_HARMONIA_CP_TX_SDR1`, `PMT_HARMONIA_CP_TX_SDR2`, and `PMT_HARMONIA_CP_TX_SDR3`. While the receive carrier phase estimates are held under the keys `PMT_HARMONIA_CP_RX_SDR1`, `PMT_HARMONIA_CP_RX_SDR2`, and `PMT_HARMONIA_CP_RX_SDR3`. Finally, the

Table 3.7: Metadata Keys for an Output PDU of the Clock Bias and Carrier Phase Estimation Block.

Metadata Key	Description
PMT_HARMONIA_CB_SDR1	Holds the clock bias estimate for SDR 1.
PMT_HARMONIA_CB_SDR2	Holds the clock bias estimate for SDR 2.
PMT_HARMONIA_CB_SDR3	Holds the clock bias estimate for SDR 1.
PMT_HARMONIA_R_SDR12	Holds the range estimate between SDR 1 and 2.
PMT_HARMONIA_R_SDR13	Holds the range estimate between SDR 1 and 3.
PMT_HARMONIA_R_SDR23	Holds the range estimate between SDR 2 and 3.
PMT_HARMONIA_CP_TX_SDR1	Holds the transmit carrier phase estimate for SDR 1.
PMT_HARMONIA_CP_TX_SDR2	Holds the transmit carrier phase estimate for SDR 2.
PMT_HARMONIA_CP_TX_SDR3	Holds the transmit carrier phase estimate for SDR 3.
PMT_HARMONIA_CP_RX_SDR1	Holds the receive carrier phase estimate for SDR 1.
PMT_HARMONIA_CP_RX_SDR2	Holds the receive carrier phase estimate for SDR 2.
PMT_HARMONIA_CP_RX_SDR3	Holds the receive carrier phase estimate for SDR 3.
clock_bias_enable	Indicates to the LFM and UHD blocks to apply the estimated clock bias estimates for timing correction.
carrier_phase_enable	Indicates to the LFM and UHD blocks to apply the estimated carrier phase estimates for phase correction.

Boolean flags `clock_bias_enable` and `carrier_phase_enable` are initialized to indicate to the LFM and UHD blocks whether the estimated clock bias and/or carrier phase corrections should be applied to the waveforms. A summary of the metadata keys is listed in Table 3.7.

Figure 3.12 demonstrates how the Time Peak Estimation blocks interface with the Clock Bias and Carrier Phase Estimation block to simulate the exchange of the time and phase estimates within the synchronization flowgraph. The Time Peak Estimation block produces time and phase estimates, which are used as inputs for estimating the remaining synchronization parameters. Similar to the clock drift estimation process, once all intermediate estimates have been collected, the block computes the final clock bias and carrier phase estimates. These are then used to update the metadata for subsequent transmitted and received signals, ensuring that each SDR in the network applies the correct compensation.

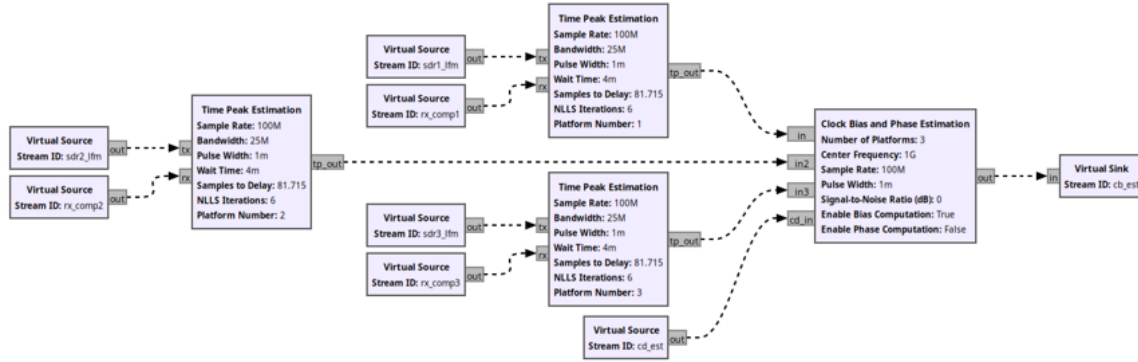


Figure 3.12: Flowgraph connection between the Time Peak Estimation blocks and the Clock Bias and Carrier Phase Estimation block.

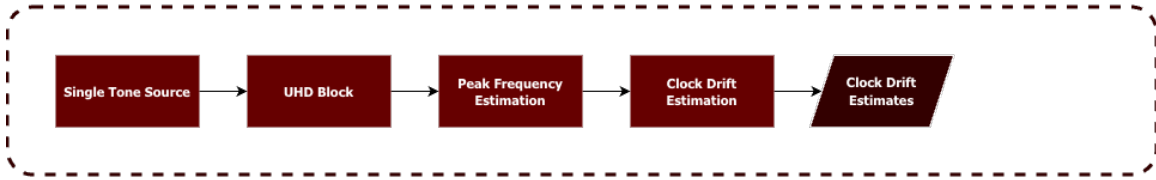
3.3 Flowgraph Implementation and Module Limitations

This section explains how the synchronization blocks are connected within the flowgraph and how each block contributes to estimating the synchronization parameters. The organization of the blocks reflects both the structure of the algorithm and several limitations encountered during development, such as the need for coarse synchronization and the constraints of the message-based data processing.

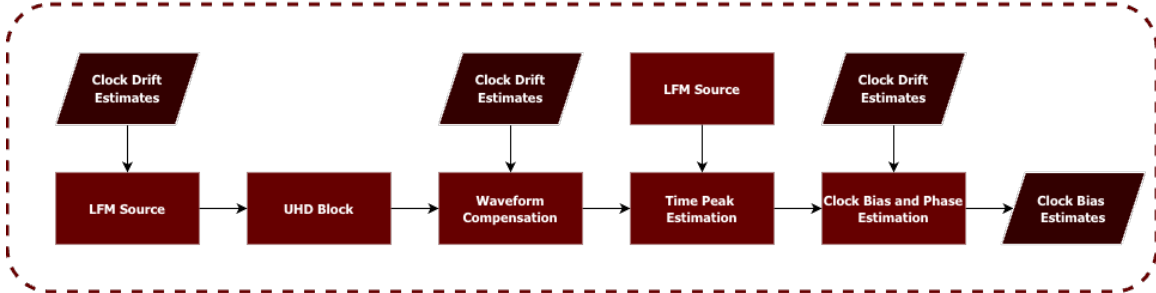
Figure 4.15 illustrates how the `gr-harmonia` blocks are arranged in a flowgraph to perform synchronization of a distributed radar system. For clarity, the diagram shows the flowgraph from the perspective of a single platform. In the full implementation, additional sets of blocks, such as the waveform generation blocks, are duplicated to produce the transmitted waveforms of the other platforms in the system. Additionally, the Frequency Peak Estimation and Time Peak Estimation blocks are duplicated twice in a three-SDR network so that each platform can perform the estimation exchange technique, as demonstrated in Figures 3.9 and 3.12.

The flowgraph begins with the Single Tone Source block, which generates a single-tone waveform and metadata and sends them to the UHD block for transmission during that platform's assigned TDMA slot. When each received waveform is processed by the UHD

Clock Drift Estimation



Clock Bias Estimation



Carrier Phase Estimation

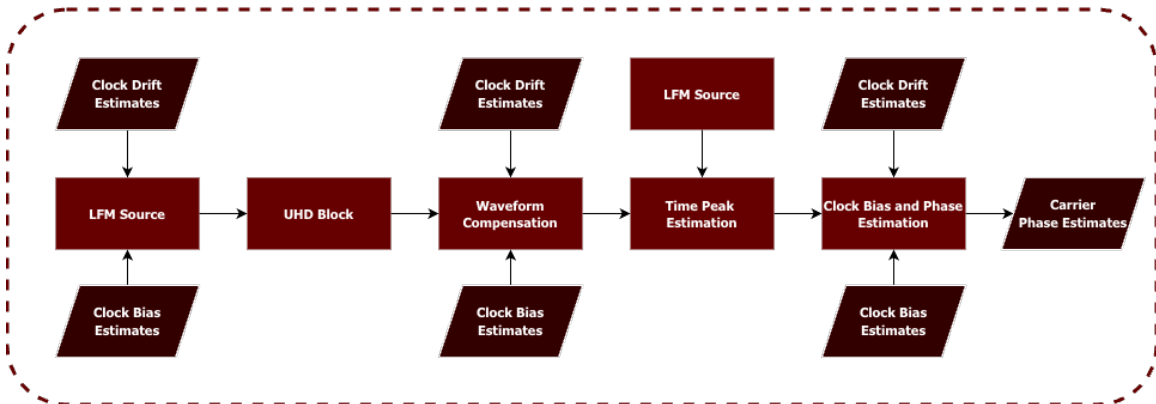


Figure 3.13: Flowgraph diagram showing how the gr-harmonia blocks are interconnected to perform synchronization in a distributed radar system.

block, it is passed to the Frequency Peak Estimation block to determine the frequency of the received signal. Each frequency estimate is then sent to the Clock Drift Estimation block along with metadata indicating the platform it came from. The Clock Drift Estimation block waits until it has received all estimates from each platform. Once all estimates are collected, it computes the clock drift for each platform and assigns metadata that identifies

which platform each clock drift value belongs to. This allows the estimates to be used correctly in the next stage.

The second stage of the flowgraph performs clock bias estimation. This stage is necessary because current coarse synchronization alone does not place the platform close enough in time for reliable phase estimation. Otherwise, the clock bias and carrier phase estimation stages could be combined into a single step. Clock bias estimation begins with the LFM Source block using the clock drift estimate to adjust the outgoing waveforms. It then includes the clock drift estimation as metadata before sending it to the UHD block for transmission. The UHD block captures the LFM's transmitted by the other platforms whenever their TDMA slots occur. All received LFM's are then passed to the Waveform Compensation block, which compensates for clock drift, and forwarded to the Time Peak Estimation block, where it is matched filtered with the ideal waveform generated by a separate LFM Source block. The Time Peak Estimation block then estimates both the time delay and phase at the matched-filter peak. These estimates are then sent to the Clock Bias and Phase Estimation block, which stores them until all estimates are collected. Since the goal of this stage is only to compute the clock bias, the parameter for clock bias estimation is only enabled. Once all estimates are available, the block computes the clock bias for each platform and assigns metadata identifying which platform each value belongs to.

Lastly, the carrier phase estimation stage is carried out using the same sequence of blocks as the clock bias stage. The key difference is that this stage incorporates the clock bias estimates, which brings the timing of each platform close enough for the phase offsets to be accurately resolved. In this stage, the Clock Bias and Phase Estimation block is configured to only calculate for carrier phase estimation. Once this step is complete, all synchronization parameters are obtained and ready to be used for future signal exchanges.

Several limitations were encountered during module development. One significant limitation stems from the coarse synchronization simulation. Since the goal is for the SDRs

not to rely on a Global Positioning System (GPS), coarse alignment was achieved by resetting the internal UHD time of each device through software. As a result, the platforms were able to synchronize within a few milliseconds of one another in the initial stages. However, using the reset function of UHD introduced significant timing and control challenges when individual UHD blocks were assigned to each SDR. To overcome this issue, a unified module was developed to centralize the coarse timing control of the radar system. Consequently, communication latency and the use of message-domain data transfer further limited both the achievable sampling rate and the amount of data that could be processed during experimentation.

Chapter 4

Experimentation and Analysis

This chapter presents the experimentation and analysis of the synchronization algorithm implemented in `gr-harmonia`. The experiments were designed to evaluate the algorithm's performance in estimating and compensating for clock drift, clock bias, and carrier phase offsets across three USRP X310 SDRs. Through a combination of loopback tests and over-the-air measurements, the objective was to quantify the module's synchronization accuracy and stability under varying operating conditions. Each section of this chapter discusses the hardware setup, calibrations, performance metrics, and test configurations, followed by the results and analysis.

4.1 Hardware Setup and Calibrations

The following section describes the hardware configuration and calibration procedures used to test the performance of `gr-harmonia` in synchronizing distributed radar networks. A description of the hardware is provided to highlight the capabilities of both the radios and the host computer. Understanding the hardware is critical for identifying sources of hardware-induced time offsets affecting the accuracy of synchronization estimation. Later in this section, the calibration process is discussed, covering cable delay measurements, front-end delay characterization, and USRP IQ balancing to ensure accurate time estima-

tion across all the SDRs.

4.1.1 Hardware Setup

The synchronization algorithm was implemented and tested using the GNU Radio software and the Ettus USRP hardware. The experimental testbed consisted of three USRP X310 devices, each equipped with two UBX-160 daughterboards to enable wide-band operation. The devices provide the required processing capability for synchronization, as shown in Table 4.1 [30]. Although the USRP X310 supports a maximum sample rate of 200 MS/s, the radios in this research operated at 100 MS/s due to UHD and computational limitations when processing multiple SDRs on a single GNU Radio block. At this operating rate, the radios achieve a timing resolution of 10 ns, which is negated by using the FFT-based fractional delay technique. Furthermore, the internal clock reference accuracy of the X310 is rated at 2.5 ppm, which, at a 1 GHz center frequency, corresponds to approximately 2.5 kHz of error. This is an important factor to consider in later analysis, as it influences clock drift. Lastly, the UBX-160 daughterboard was selected for its 160 MHz instantaneous bandwidth and its wide tuning range from 10 MHz to 6 GHz, allowing experiments to be conducted across a variety of carrier frequencies [24].

The radios were connected to a host computer via 10 Gigabit Ethernet (10GbE) to ensure high-rate data exchange for synchronization. The host computer, running Ubuntu 24.04, was built on a Gigabyte MC62-G40 motherboard with an AMD Ryzen Threadripper PRO 5955WX CPU and four 32 GB DDR4-3200 MHz RAM modules. This configuration provided the computational power and ports needed to run three X310s simultaneously within GNU Radio and the host computer. Additionally, a Siglent SDG6000X signal generator was used to supply a 10 MHz reference clock to all SDRs for testing and debugging of clock drift, clock bias, and carrier phase.

Table 4.1: Important RF specification of the Ettus USRP X310 and UBX-160 daughter-board. Data from [24], [30].

Spec	Typ.	Unit
USRP X310 Motherboard		
ADC Sample Rate (max)	200	MS/s
ADC Resolution	14	bits
DAC Sample Rate (max)	800	MS/s
DAC Resolution	16	bits
Host Sample Rate (16b)	200	MS/s
Internal Reference Accuracy	2.5	ppm
UBX-160 RF Daughterboard		
Frequency Range	10-6000	MHz
Instantaneous Bandwidth	160	MHz

4.1.2 Hardware Calibration

Due to the sensitivity of time delay estimation at high frequencies, several calibration procedures were conducted to account for hardware-induced delays. The primary sources of propagation delay were identified as the front-end transmit and receive paths, RF cables, and antennas. Cable delay was first measured using a Keysight N5225A-200 PNA network analyzer at 1 GHz, yielding a time delay of approximately 1.44 ns per 12-inch cable. This measured delay was used to correct the timing offset in the following measurement of the front-end delay.

Each USRP X310s was then configured in loopback mode, with each SDR transmitting and receiving its own LFM signal. Then, in MATLAB, the signal was processed using a matched filter and sinc-based NL-LS to estimate the front-end delay with precision. The resulting front-end delay was measured to be approximately 817.15 ns at 1 GHz after accounting for the cable delay. Similarly, to determine the time delay introduced by the monopole antennas used in the OTA experiments, two SDRs were configured at a set distance of 1 ft, shown in Figure 4.1. Using the network analyzer, the resulting time delay

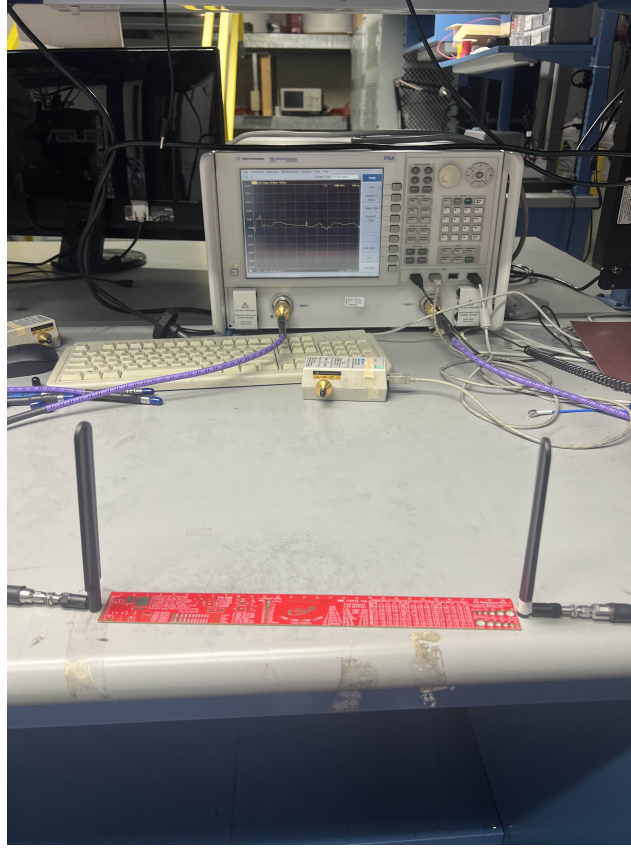


Figure 4.1: Monopole antenna calibration setup.

introduced by each antenna was approximately 1.33 ns, after accounting for propagation delay. Additionally, USRP IQ calibration was performed to minimize I/Q imbalance and DC offset, which can introduce phase and amplitude errors in the received signal [23]. These were conducted using pre-defined calibration routines provided by the UHD framework.

4.2 Experimental Setups and Performance Analysis

The following section presents the experimental setups and Cramér-Rao Lower Bound (CRLB) to evaluate the performance of `gr-harmonia` under each experiment. Two primary configurations were designed: a loopback setup to isolate from external noise and an

OTA setup to assess full synchronization accuracy in a wireless environment. The loopback setup was primarily developed to validate the performance of clock drift estimation, while the OTA setup mainly tested the performance of clock bias and carrier phase estimation. The following subsections present the block diagrams, parameters, and associated CRLB for each experiment.

4.2.1 Cramér-Rao Lower Bound

The Cramér-Rao Lower Bound establishes the theoretical lower limit on the variance of an unbiased estimator. In other words, it defines the best possible accuracy that an estimator can achieve when estimating a given parameter. The CRLB depends on the signal-to-noise ratio (SNR) and other variables, depending on the parameter being estimated. Since the final estimates are refined using NL-LS, a direct calculation of the CRLB for clock drift, clock bias, and carrier phase is not available. Instead, the performance can be evaluated by examining the CRLBs of the frequency, time, and phase estimators and demonstrating, through tests, that these estimators closely follow these bounds across a set SNR range. This subsection introduces the CRLB equations for time, frequency, and phase, and discusses how they are used to evaluate the performance of clock drift, clock bias, and carrier phase estimation. All CRLB equations used in this work are taken from [16].

To assess the performance of clock drift estimation, the CRLB of frequency must first be defined, which is given by

$$\sigma_f^2 \geq \frac{3}{2\pi^2 T_p^3 f_s \cdot \text{SNR} \left(1 - \left(\frac{1}{T_p f_s}\right)^2\right)}. \quad (4.1)$$

Figure 4.2 shows the frequency CRLB for the clock drift experiment, computed using a pulse width of $T_p = 300 \mu s$ and a sampling frequency of 100 MHz, over an SNR range

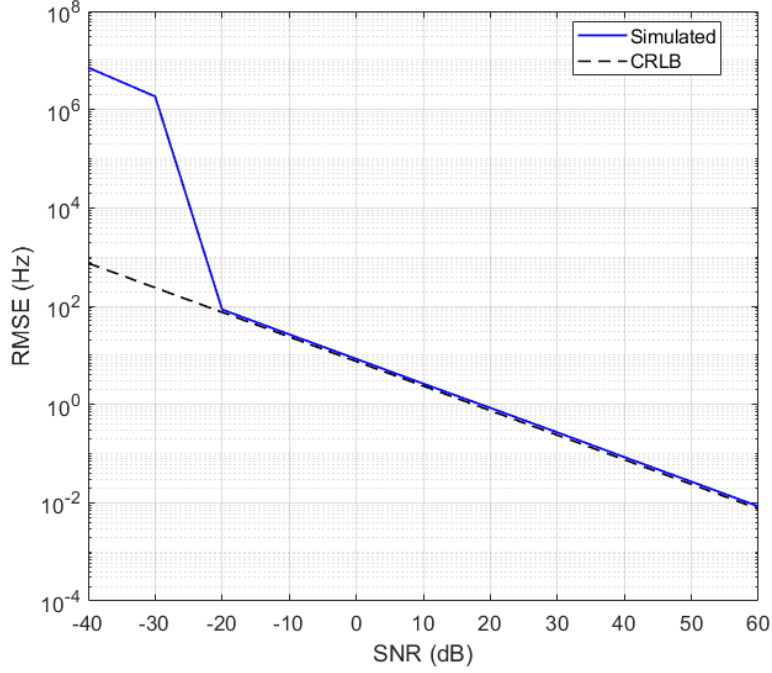


Figure 4.2: Frequency Cramér-Rao Lower Bound with cabled experiment parameters.

of -40 dB to 60 dB. Additionally, Figure 4.3 presents the frequency CRLB for the OTA experiment, calculated with a pulse width of $T_p = 9 \text{ ms}$ and a sampling frequency of 100 MHz. These will be used for analysis later in this chapter.

The variances of the frequency estimates between SDRs, denoted as $V_{i,j}^f$, can be obtained using (4.1) and arranged into a covariance matrix shown as

$$C_f = \text{diag}\left(\left[V_{1,2}^f, V_{1,3}^f, \dots, V_{1,N}^f, V_{2,1}^f, V_{2,3}^f, \dots, V_{N-1,N}^f, \varepsilon_f\right]\right), \quad (4.2)$$

where ε_f is an arbitrary small value to account for $\alpha_1 = 1$.

The error covariance of the clock drift estimate can then be calculated by

$$\Sigma_\alpha = (A_f^T C_f^{-1} A_f)^{-1}, \quad (4.3)$$

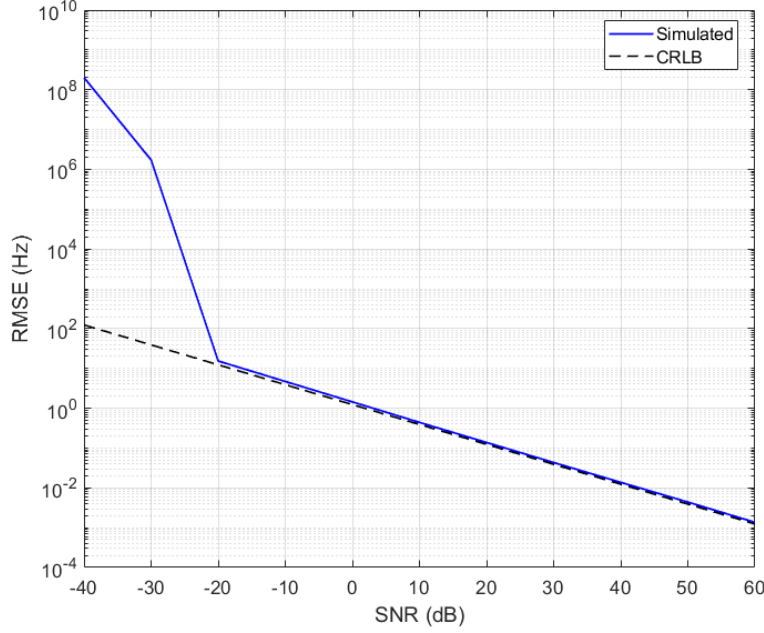


Figure 4.3: Frequency Cramér-Rao Lower Bound with OTA experiment parameters.

where A_f is a $N^2 - N + 1 \times N$ matrix containing the coefficients of the system of equation in (2.19). The clock drift CRLB for the two experiments is presented in Figures 4.4 and 4.5, which illustrate how longer pulses can reduce the RMSE of clock drift.

Since clock bias corresponds to the timing offsets of the SDRs, its performance can be evaluated through the CRLB of the time estimator. The CRLB for time delay is given by

$$\sigma_\tau^2 \geq \frac{3}{2\pi^2\beta^2 f_s T_p \cdot \text{SNR}}. \quad (4.4)$$

Given the parameters $\beta = 25 \text{ MHz}$, $T_p = 9 \text{ ms}$, and $f_s = 100 \text{ MHz}$, the CRLB for time estimation is plotted in Figure 4.6.

Using (4.4) to compute the variance of the time measurements, denoted by $V_{i,j}^\tau$, the variance of the clock bias offset estimate is given by

$$V_{i,j}^\phi = \frac{1}{4} (V_{i,j}^\tau + V_{j,i}^\tau). \quad (4.5)$$

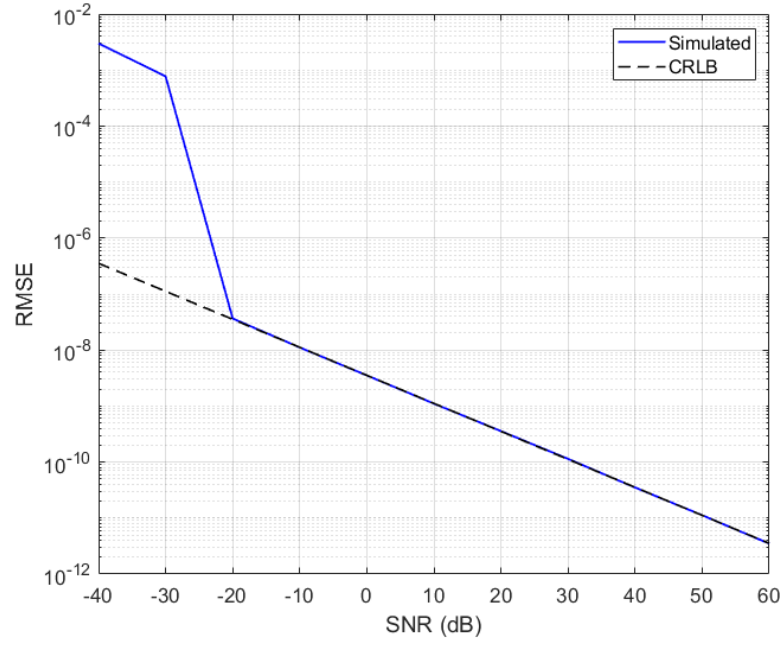


Figure 4.4: Clock drift Cramér-Rao Lower Bound with cabled experiment parameters.

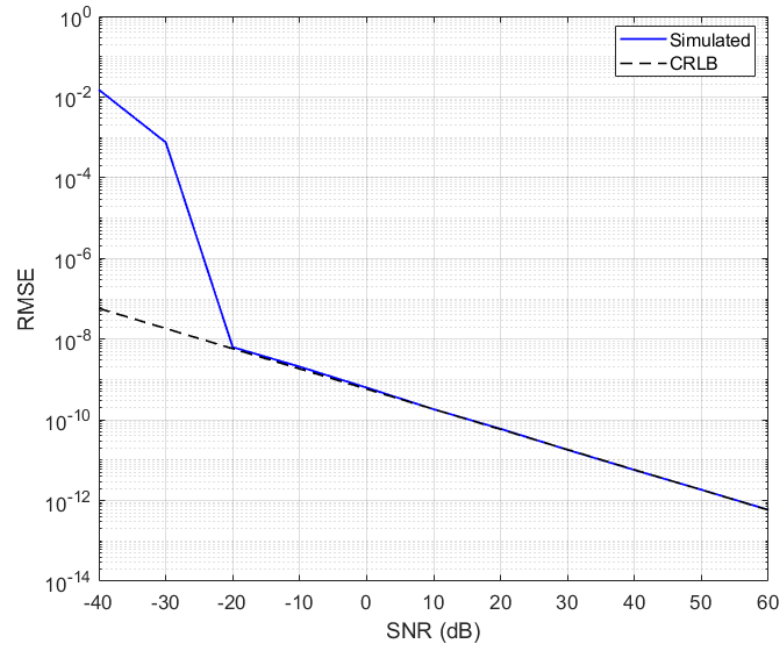


Figure 4.5: Clock drift Cramér-Rao Lower Bound with OTA experiment parameters.

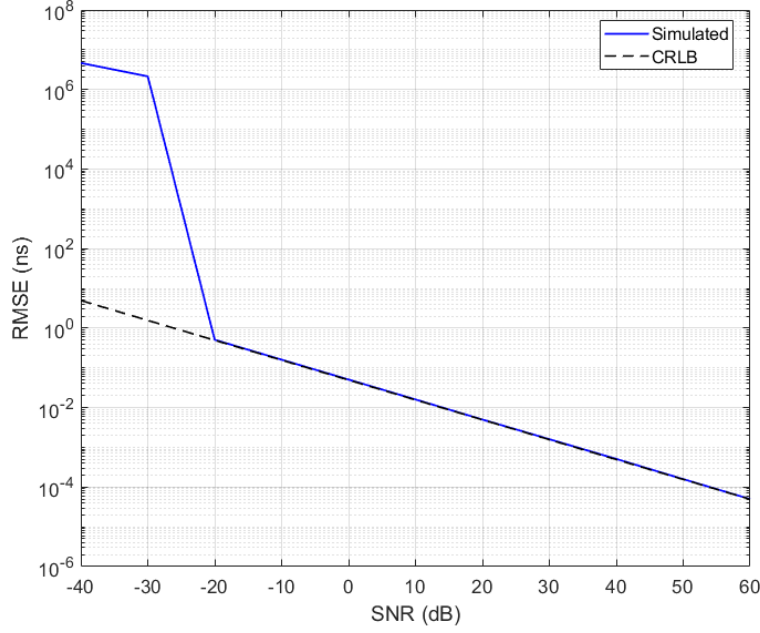


Figure 4.6: Time Cramér-Rao Lower Bound.

These variances can then be organized into the covariance matrix defined by

$$C_{\phi} = \text{diag}\left(\left[V_{1,2}^{\phi}, V_{1,3}^{\phi}, \dots, V_{1,N}^{\phi}, V_{2,1}^{\phi}, V_{2,3}^{\phi}, \dots, V_{N-1,N}^{\phi}, \varepsilon_{\phi}\right]\right), \quad (4.6)$$

where similar to ε_f , ε_{ϕ} is chosen to be a value close to zero to describe the variance in the solution to $\phi_1 = 0$. However, since clock bias is difficult to measure due to its inconsistency across runs, range will be used instead to assess the experiment's performance. The CRLB of the range can be computed by

$$V_{i,j}^R = \frac{c^2}{4} (V_{i,j}^{\tau} + V_{j,i}^{\tau}). \quad (4.7)$$

The range variances can then be put into a covariance matrix given by

$$\Sigma_R = \text{diag}\left(\left[V_{1,2}^R, V_{1,3}^R, \dots, V_{1,N}^R, V_{2,1}^R, V_{2,3}^R, \dots, V_{N-1,N}^R\right]\right). \quad (4.8)$$

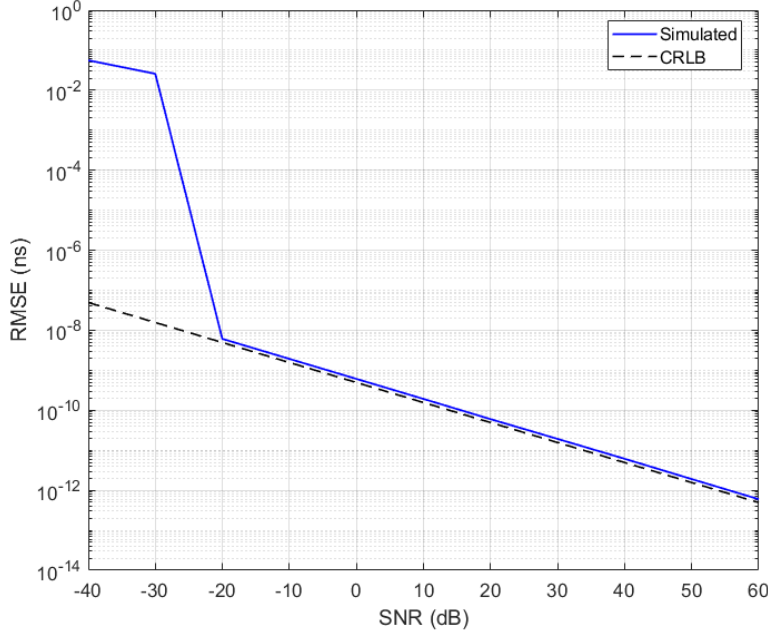


Figure 4.7: Clock bias Cramér-Rao Lower Bound.

Figures 4.7 and 4.8 illustrate the error covariances for clock bias and range, respectively, using the same parameters as the time CRLB.

Lastly, the CRLB corresponding to phase estimation can be expressed as

$$\sigma_{\gamma}^2 \geq \frac{1}{2T_p f_s \cdot \text{SNR}}. \quad (4.9)$$

Figure 4.9 shows the CRLB for phase estimation using a pulse width of $T_p = 50 \mu s$ and a sampling frequency of $f_s = 100 \text{ MHz}$.

Let the variance of the carrier phase difference be calculated by

$$V_{i,j}^{err} = V_{i,j}^{\gamma} + (2\pi f^c)^2 \left(\frac{V_{i,j}^{\tau}}{2} + \frac{V_{j,i}^{\tau}}{2} \right), \quad (4.10)$$

where $V_{i,j}^{\gamma}$ is the variance of the phase computed by (4.9) and f^c is the center frequency.

Then the variance can be arranged into a covariance matrix defined by

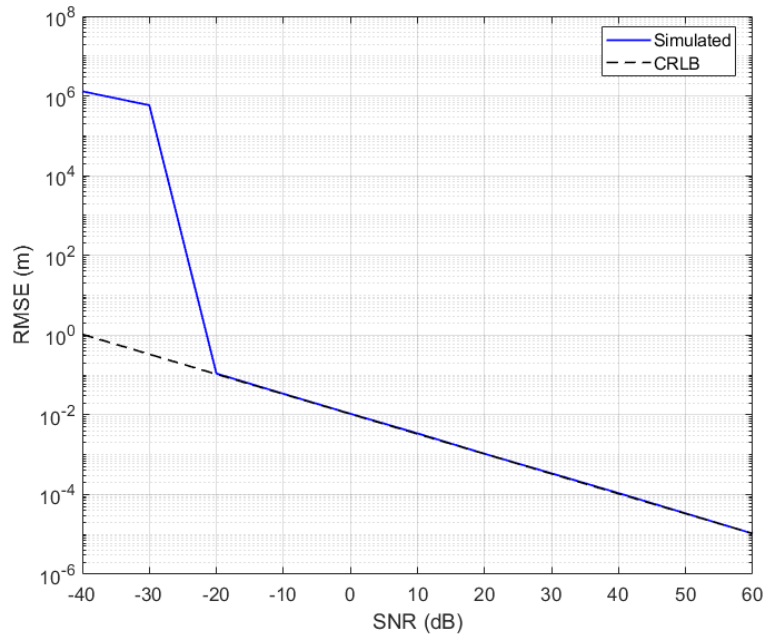


Figure 4.8: Range Cramér-Rao Lower Bound.

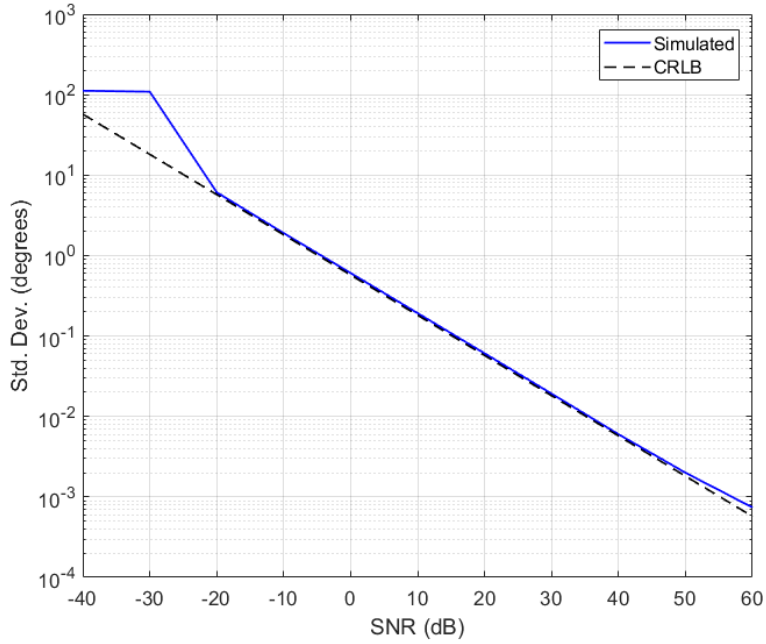


Figure 4.9: Phase Cramér-Rao Lower Bound.

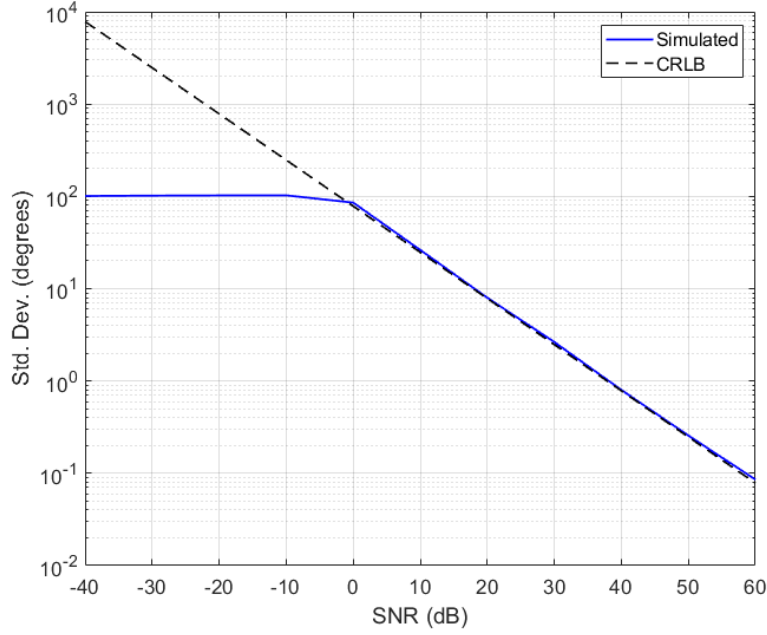


Figure 4.10: Carrier phase Cramér-Rao Lower Bound.

$$C_{err} = \text{diag}\left(\left[V_{f_{1,2}}^{err}, V_{f_{1,3}}^{err}, \dots, V_{f_{1,N}}^{err}, V_{f_{2,1}}^{err}, V_{f_{2,3}}^{err}, \dots, V_{f_{N-1,N}}^{err}, \varepsilon_{err}\right]\right), \quad (4.11)$$

where ε_{err} is an arbitrarily low value to account for γ_1^{tx} .

Then, the error covariance of the carrier phase can be computed by

$$\Sigma_\gamma = (A_\gamma^T C_{err}^{-1} A_\gamma)^{-1}. \quad (4.12)$$

Figure 4.10 presents the CRLB for carrier phase, using the same parameter used in the phase CRLB given in Figure 4.9.

The CRLB analysis provides a theoretical benchmark for evaluating the performance of the synchronization estimates. Through the computation of the frequency, time, range, and phase estimation bounds, the expected lower limits of each estimation error variance

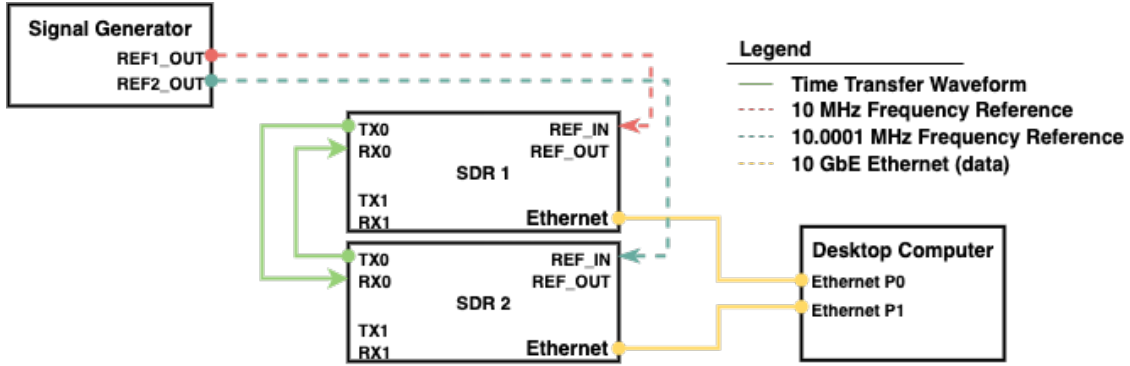


Figure 4.11: Block diagram of cabled two-node system setup used to assess clock drift estimation.

were established. The next subsection focuses on the experimental setups used to validate the clock drift estimation.

4.2.2 Clock Drift Estimation in A Cabled Two-Node System

The clock drift experiment was designed with two SDRs driven by separate reference signals generated by a signal generator. One node received a 10 MHz signal, while the other was given a 10.0001 MHz signal to simulate a 10 ppm clock drift between the two radios. A block diagram of the setup is shown in Figure 4.11, and the parameters are given in Table 4.2. The sample rate was set at 100 MS/s, as higher frequencies introduced latency that caused the module to fail. The center frequency was chosen to be 1 GHz to allow for performance evaluation of the clock drift blocks at high carrier frequencies. The baseband frequency, pulse width, capture time, wait time, and TDMA time were chosen arbitrarily to ensure that the transmitted signals were fully captured within each measurement window.

The physical experimental setup is shown in Figure 4.12, illustrating the two SDRs, signal generator, and cabling arrangement used in this experiment. RF attenuators were included, as recommended by Ettus Research documents, to prevent damage to the devices from high-power transmissions [23]. The time delay introduced by the attenuators was

Table 4.2: Parameters for the Two-Node Clock Drift Estimation Experiment.

Parameter	Value
Sample rate	100 MS/s
Center frequency	1 GHz
Baseband frequency	10 MHz
Pulse width	300 μ s
Capture time	600 μ s
Wait time	200 μ s
TDMA time	1 ms

found to be negligible. The GNU Radio flowgraph was initiated to transmit and receive the pulsed single-tone waveform while applying the clock drift estimation algorithm. The 10 MHz reference signal drove SDR 1, while SDR 2 received the 10.0001 MHz signal. The estimated drift values were extracted using the `gr-plasma` PDU File Sink block for analysis. A total of 100 experiments were conducted to evaluate the standard deviation of the clock drift estimation. The received synchronization tone had an approximate SNR of 30 dB, determined by measuring a sample of the received signal power and noise power and taking their ratio. As shown in Figure 4.4, the CRLB establishes a theoretical lower limit of approximately 1.17×10^{-10} for the clock drift standard deviation. In comparison, the resulting estimation produced a standard deviation of 4.0420×10^{-10} and a mean error of 1.5371×10^{-10} . Although the results lie above the theoretical limit, they remain within an order of magnitude. This demonstrates that the `gr-harmonia` module is capable of accurately estimating clock drift. Figure 4.13 illustrates two LFMs sent between the devices to validate the clock drift compensation process. The results reveal that the compensated waveforms match at the 10 MHz tone; however, the elevated sidelobes indicate that some residual misalignment remains.

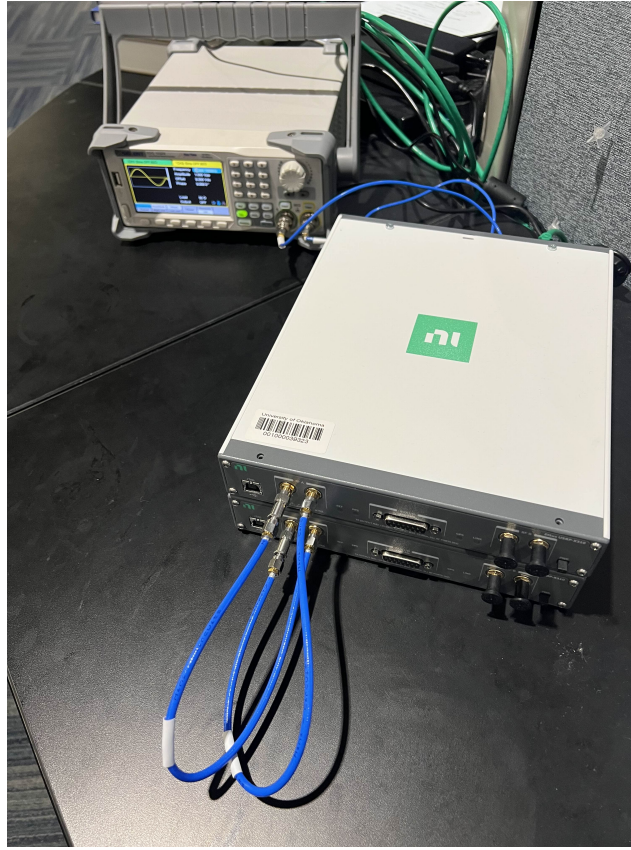


Figure 4.12: Two-node cabled system setup for evaluating clock drift estimation performance.

4.2.3 Clock Bias and Carrier Phase Estimation in Over-the-Air System Experiments

For the over-the-air experimentation, the distributed radar system was designed with three SDRs using a common 10 MHz reference clock generated by a signal generator. The reference clock was daisy-chained between radios via the reference in/out ports to provide a common frequency for the system. This is necessary for accurate clock bias and carrier phase estimation, which will be discussed later in this subsection. All over-the-air measurements were conducted in an anechoic chamber to mitigate multipathing interference and ensure a controlled testing environment. A block diagram of the setup is shown in

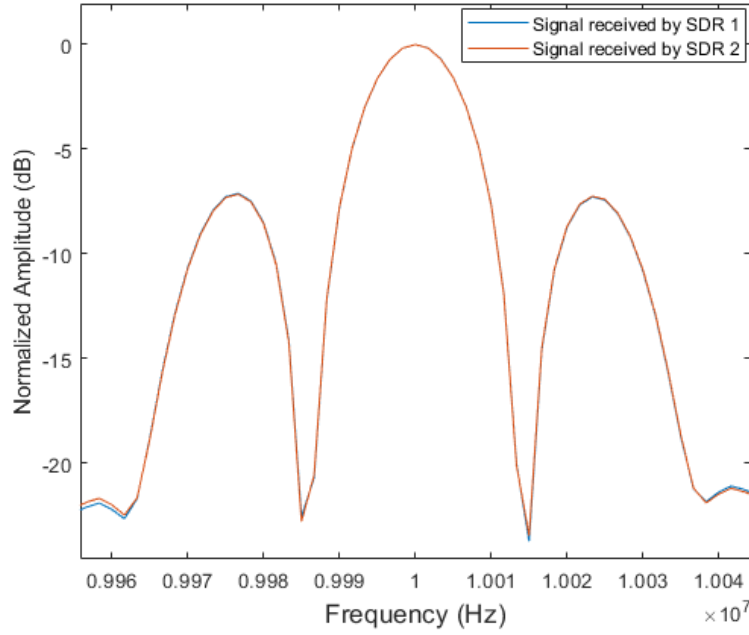


Figure 4.13: FFT of the received signals from SDR 1 and 2 after clock drift compensation.

Figure 4.14, and the parameters are given in Table 4.3. Similar to the previous experiment, the sample rate was set at 100 MS/s to mitigate latency. Each platform was tuned to a 1 GHz center frequency, and the baseband frequency for the waveforms was set to 10 MHz. The pulse width, capture time, wait time, TDMA time were chosen arbitrarily to ensure each platform transmitted independently and received by the other. Once clock bias is estimated, the second round of timing parameters (pulse width 2, capture time 2, wait time 2, and TDMA time 2) is implemented for carrier phase estimation. The duration of these parameters is intentionally shorter because phase changes occur more rapidly at higher carrier frequencies, making it important to exchange signals within a shorter time interval.

Figure 4.15 shows the physical setup of the OTA experiments inside the anechoic chamber. The three SDR platforms and signal generator were arranged to maintain direct line-of-sight (LOS) between antennas. To further reduce the potential for multipathing interference, clear empty bins were used to elevate the radios above the chamber floor. This

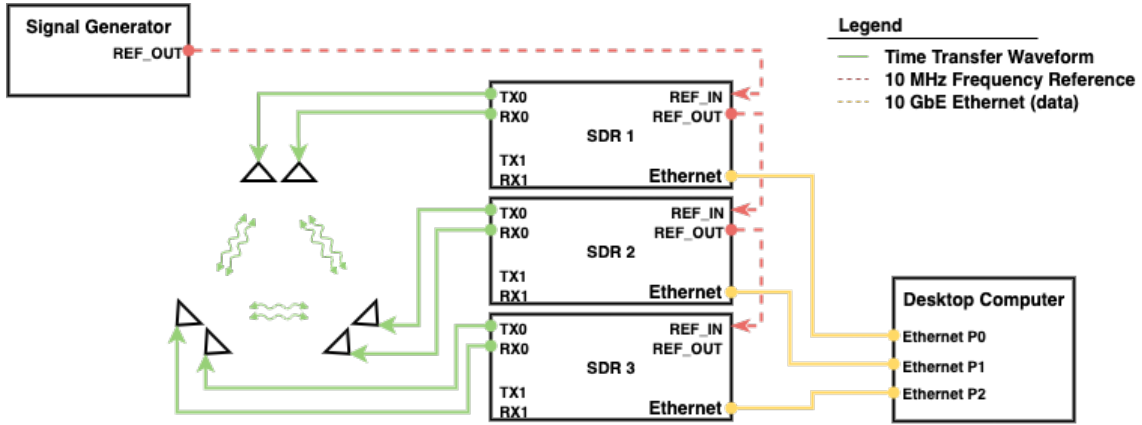


Figure 4.14: Block diagram of over-the-air system synchronization setup used to assess synchronization parameter estimation.

Table 4.3: Parameters for the Full Over-the-Air Synchronization Experiment.

Parameter	Value
Sample rate	100 MS/s
Center frequency	1 GHz
Baseband frequency	10 MHz
Bandwidth	25 MHz
Pulse width	1 ms
Pulse width 2	50 μ s
Capture time	9 ms
Capture time 2	100 μ s
Wait time	4 ms
Wait time 2	10 μ s
TDMA time	15 ms
TDMA time 2	150 μ s

configuration helped eliminate reflections from the ground, ensuring that only the direct LOS signals were measured during the experiments.

Two sets of OTA experiments were conducted: one using the internal reference clocks of the SDRs and another using the external 10 MHz reference clock distributed from the signal generator. The first experiment was initially expected to achieve coherent operation; however, the clock drift estimates were not accurate enough. As a result, the lack of coherent frequency operation between radars resulted in unreliable clock bias and carrier

Table 4.4: Clock Drift Estimation Results.

SDR	Standard Deviation
2	7.17 ppb
3	6.77 ppb

phase estimation. The second experiment was designed with the external 10 MHz reference clock to simulate an ideal clock drift estimation scenario, providing a reference case for evaluating the performance of clock bias and carrier phase.

The received waveforms exhibited an approximate SNR of 20-30 dB, determined from sampled measurements of their signal and noise power. For these OTA parameters at the highest SNR, the corresponding CRLB established a theoretical limit of 1.835×10^{-11} , as shown in Figure 4.5. However, the experimental results in Table 4.4 reveal that the clock drift was estimated with standard deviations of 7.17 ppb and 6.77 ppb for SDR 2 and SDR 3, respectively. These values are substantially higher than the theoretical limit and are likely influenced by multipathing interference. At a carrier frequency at 1 GHz, a standard deviation this high is insufficient for accurate estimation of clock bias, which is confirmed by the results in Table 4.5.

The impact of the degraded clock drift accuracy can be seen in the range estimation. The CRLB standard deviation of range at an SNR of 30 dB is approximately 0.033 cm, as shown in Figure 4.8. The experimental results in Table 4.5 show significantly large errors when the internal reference clock is used. Switching to the external 10 MHz reference clock notably improves the performance, reducing the range standard deviation to between 3.22 cm and 4.48 cm. This level of error is sufficient for carrier phase estimation, since at a 1 GHz carrier frequency, the wavelength is approximately 30 cm. As a result, the corresponding phase change is minimal and does not significantly degrade overall carrier phase estimation performance.

To demonstrate beamforming performance and phase coherency with an external ref-



Figure 4.15: Over-the-air synchronization setup in an anechoic chamber for evaluating synchronization parameter estimation.

Table 4.5: Range and Clock Bias Estimation Results.

Test	Dev 1 $\rightarrow$ Dev 2	Dev 1 $\rightarrow$ Dev 3	Dev 2 $\rightarrow$ Dev 3
	Standard Deviation		
Internal Clock	26.18 cm (0.87 ns)	45.73 cm (1.52 ns)	61.57 cm (2.05 ns)
External Clock	3.22 cm (0.10 ns)	3.92 cm (0.13 ns)	4.48 cm (0.15 ns)

erence clock, nodes 2 and 3 were configured to transmit toward node 1. Both nodes were assigned a $50 \mu s$ LFM that incorporated compensation for clock bias and range offsets. Given three TDMA time slots, node 2 was set to transmit during the first and second slots, while node 3 was assigned the second and third. This configuration resulted in the signal shown in Figure 4.16. An averaging filter with a 200-sample window was then applied to the received waveform to compute the power. Afterwards, the power of each pulse was

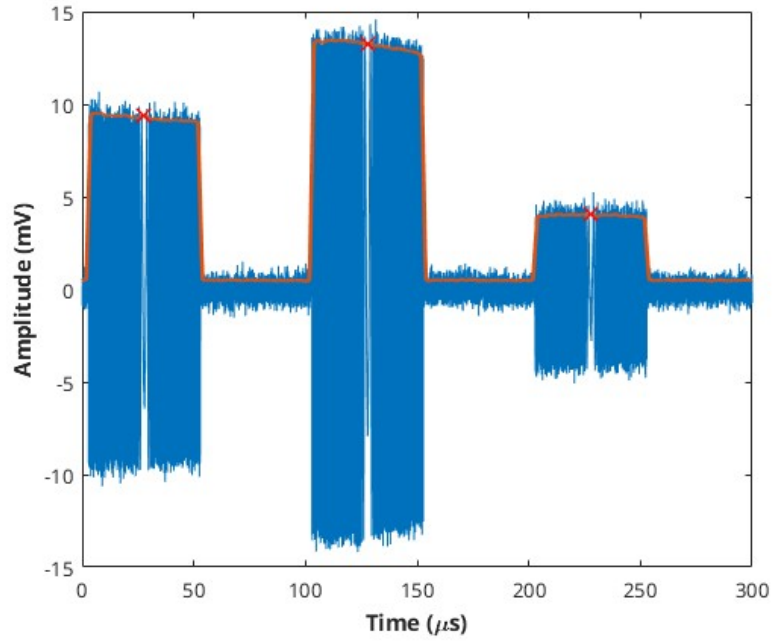


Figure 4.16: Over-the-air beamforming experiment in which SDR 1 transmits individually, then SDRs 1 and 2 transmit together, and finally SDR 2 transmits individually to SDR 3 to assess phase coherency.

sampled at the midpoint of each LFM. Using these sampled values, the ratio of the combined transmit power, and the sum of the single-transmit powers yielded a coherent gain estimate of 98.62% of the maximum possible coherent gain.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

The `gr-harmonia` module presents a fully digitized synchronization framework for SDRs operating within a distributed radar network using GNU Radio. The module functions entirely in the message domain, where all signal processing blocks communicate through PDUs, enabling metadata handling and debugging across the synchronization blocks. In its current stage of development, the module assumes stationary platforms and demonstrates the capability to estimate the synchronization parameters, which include clock drift, clock bias, and carrier phase.

The experimental results demonstrate the performance of these estimators in different environments. Clock drift is estimated to be within approximately 6-7 ppb. Clock bias estimation achieves sub-nanosecond accuracy when an external reference clock is used, validating the effectiveness of the synchronization algorithm's time estimation. Additionally, the system achieves a maximum gain of 98.62% during beamforming experiments, confirming that the phase corrections by the module are sufficiently accurate to support coherent processing across multiple SDRs. These results establish a foundation for synchronization in distributed radar systems for further development of the module.

5.2 Future Work

Future developments of `gr-harmonia` will focus on improving the synchronization framework by incorporating additional estimation techniques and refining existing blocks. The first enhancement is the incorporation of a Kalman filter into the synchronization loop [31]. The Kalman filter would help refine clock drift estimates by choosing between predicted values and observed measurements. This would allow for the module to synchronize a radar system without assistance from outside references fully. In addition, integrating SNR estimation with weighted least-squares (WLS) can improve estimation performance under non-ideal noise conditions [15]. This would allow the radios to rely more on measurements with stronger SNR, improving the overall estimation of the synchronization parameters and reducing sensitivity to noisy or unreliable measurements.

Furthermore, `gr-harmonia` will be extended to support platform motion. This includes adding compensation for relative velocity and Doppler shifts, which become increasingly important when operating any mobile platform. By incorporating velocity estimation and compensation into the module, it will be capable of supporting distributed radar networks and enabling coherent processing across SDRs in motion. These advancements will allow the `gr-harmonia` module to be applied in the real world.

Implementing these capabilities requires substantial restructuring of the module's architecture. In particular, the UHD block has to be redesigned to operate on a single SDR, ensuring that each platform can run independently on its own computer. To accomplish this, each radar node must reliably identify itself. This can be handled by assigning a platform ID through the UHD block's metadata, allowing each SDR to determine which node it represents and when to transmit and receive. Additionally, a QPSK communication block will be required to allow each SDR to exchange its estimates with the others. This addition will require modifying the clock drift, clock bias, and carrier phase estimation blocks so

they can receive these estimates and correctly organize them by platform ID. All blocks will also need a scalable array structure and an input parameter specifying the number of platforms, allowing the system to accommodate additional nodes. Together, these changes provide the foundation for synchronizing a mobile, distributed radar system.

References

- [1] T. Comberiate, K. Zilevu, J. Hodkin, and J. Nanzer, “Distributed transmit beamforming on mobile platforms using high-accuracy microwave wireless positioning,” 98291S, May 2016.
- [2] E. Fishler, A. Haimovich, R. S. Blum, L. J. Cimini, D. Chizhik, and R. A. Valenzuela, “Spatial diversity in radars-models and detection performance,” *IEEE Trans. Signal Process.*, vol. 54, no. 3, pp. 823–838, Mar. 2006.
- [3] C. Baker and A. Hume, “Netted radar sensing,” *IEEE Aerosp. Electron. Syst. Mag.*, vol. 18, no. 2, pp. 3–6, Feb. 2003.
- [4] A. M. Haimovich, R. S. Blum, and L. J. Cimini, “Mimo radar with widely separated antennas,” *IEEE Signal Process. Mag.*, vol. 25, no. 1, pp. 116–129, Jan. 2008.
- [5] J. Liang, Q. Liang, and Z. Zhou, “Radar sensor network design and optimization for blind speed alleviation,” in *2007 IEEE Wirel. Commun. Netw. Conf.*, Mar. 2007, pp. 2643–2647.
- [6] H. Ochiai, P. Mitran, H. V. Poor, and V. Tarokh, “Collaborative beamforming for distributed wireless ad hoc sensor networks,” *IEEE Trans. Signal Process.*, vol. 53, no. 11, pp. 4110–4124, Nov. 2005.

- [7] H. Wymeersch, J. Lien, and M. Z. Win, “Cooperative localization in wireless networks,” *Proceedings of the IEEE*, vol. 97, no. 2, pp. 427–450, 2009. DOI: 10.1109/JPROC.2008.2008853.
- [8] R. Feger, A. Haderer, and A. Stelzer, “Experimental verification of a 77-ghz synthetic aperture radar system for automotive applications,” in *2017 IEEE MTT-Int. Conf. Microw. Intell. Mobil. ICMIM*, Mar. 2017, pp. 111–114.
- [9] C. Waldschmidt, J. Hasch, and W. Menzel, “Automotive radar — from first efforts to future systems,” *IEEE Journal of Microwaves*, vol. 1, pp. 135–148, Jan. 2021. DOI: 10.1109/JMW.2020.3033616.
- [10] R. H. Kenney, R. E. Jarvis, H. H. Sigmarsson, and J. W. McDaniel, “Efficient time-domain tuning of microwave filters using concepts from the unscented kalman filter,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 72, no. 4, pp. 2391–2413, 2024. DOI: 10.1109/TMTT.2023.3314115.
- [11] R. H. Kenney, K. Konyalioglu, M. Yeary, H. H. Sigmarsson, and J. W. McDaniel, “An all-cots high sampling frequency pulse-doppler imaging radar,” in *2020 IEEE Radar Conference*, Sep. 2020, pp. 1–6.
- [12] J. M. Merlo and J. A. Nanzer, “Wireless time and phase alignment for wideband beamforming in distributed phased arrays,” in *2023 IEEE International Symposium on Antennas and Propagation and USNC-URSI Radio Science Meeting (USNC-URSI)*, 2023, pp. 365–366. DOI: 10.1109/USNC-URSI52151.2023.10238024.

- [13] W. Wang, C. Ding, and X. Liang, "Time and phase synchronisation via direct-path signal for bistatic synthetic aperture radar systems," *IET Radar, Sonar Navigation*, vol. 2, pp. 1–11, 1 2008. DOI: 10.1049/iet-rsn:20060097.
- [14] S. Mghabghab and J. A. Nanzer, "A self-mixing receiver for wireless frequency synchronization in coherent distributed arrays," in *2020 IEEE/MTT-S International Microwave Symposium*, 2020, pp. 1137–1140. DOI: 10.1109/IMS30576.2020.9223809.
- [15] R. Kenney, "An approach to simultaneous wireless synchronization and navigation for mobile distributed networks of radar systems," Ph.D. Dissertation, University of Oklahoma, Norman, Oklahoma, 2024.
- [16] R. H. Kenney, J. G. Metcalf, and J. W. McDaniel, "Concept and theoretical performance analysis for decentralised digital synchronisation in distributed radar sensor networks," *IET radar, sonar navigation.*, vol. 19, no. 1, 2024.
- [17] R. H. Kenney and J. W. McDaniel, "Cooperative navigation of mobile radar sensors using time-of-arrival measurements and the unscented kalman filter," *IEEE Transactions on Radar Systems*, vol. 1, pp. 435–447, 2023. DOI: 10.1109/TRS.2023.3310862.
- [18] S. Prager, M. S. Haynes, and M. Moghaddam, "Wireless subnanosecond RF synchronization for distributed ultrawideband software-defined radar networks," *IEEE Transactions on Microwave Theory and Techniques*, vol. 68, no. 11, pp. 4787–4804, Nov. 2020.

- [19] J. M. Merlo, S. Wagner, J. Lancaster, and J. A. Nanzer, “Fully wireless coherent distributed phased array system for networked radar applications,” *IEEE Microwave and Wireless Technology Letters*, vol. 34, no. 6, pp. 837–840, 2024.
- [20] D. Pauluzzi and N. Beaulieu, “A comparison of snr estimation techniques for the awgn channel,” *IEEE Transactions on Communications*, vol. 48, no. 10, pp. 1681–1691, 2000. DOI: 10.1109/26.871393.
- [21] S. R. Mghabghab and J. A. Nanzer, “High accuracy adaptive microwave ranging using snr-based perception for coherent distributed antenna arrays,” 12, vol. 67, Dec. 2020, pp. 5540–5549.
- [22] N. Shandi, J. M. Merlo, and J. A. Nanzer, “Decentralized picosecond synchronization for distributed wireless systems,” *IEEE Transactions on Communications*, vol. 73, no. 6, pp. 4425–4438, 2024. DOI: 10.1109/TCOMM.2024.3480993.
- [23] E. K. Base, *X300/x310 — ettus knowledge base*, 2025.
- [24] Ettus Research, *Ubx-160 daughterboard specifications*, https://kb.ettus.com/UBX_160_Daughterboard, National Instruments, 2020.
- [25] *Gnu radio wiki*, GNU Radio, 2025.
- [26] S. Flandermeyer, *gr-plasma: Gnu radio module for plasma radar signal processing*, <https://github.com/ShaneFlandermeyer/gr-plasma>, Accessed: 2025-06-14, 2024.
- [27] A. V. Oppenheim, R. W. Schaffer, and J. R. Buck, *Discrete-time signal processing*, 3rd. Prentice Hall, 2010, pp. 163–165.

- [28] S. Flandermeyer, R. Mattingly, and J. Metcalf, “Gr-plasma: A new gnu radio-based tool for software-defined radar,” *Proceedings of the GNU Radio Conference*, vol. 7, no. 1, 2022.
- [29] C. Kieu, R. Kenney, and J. Metcalf, “Gr:harmonia: A gnu radio-based tool for synchronization of software-defined radios in a distributed radar network,” *Proceedings of the GNU Radio Conference*, vol. 10, no. 1, 2025.
- [30] Ettus Research, *Usrcp x310 hardware specifications*, <https://www.ettus.com/all-products/x310-kit/>, National Instruments, 2020.
- [31] R. H. Kenney, K. Konyalioglu, M. Yeary, H. H. Sigmarsson, and J. W. McDaniel, “An all-cots high sampling frequency pulse-doppler imaging radar,” in *2020 IEEE Radar Conference*, 2020, pp. 1–6. DOI: 10.1109/RadarConf2043947.2020.9266708.

Appendix A

Acronyms

ADC Analog-to-Digital Converter

API Application Programming Interface

CFAR Constant False Alarm Rate

CRLB Cramér-Rao Lower Bound

DAC Digital-to-Analog Converter

DDC Digital Down-Converter

DFT Discrete Fourier Transform

DUC Digital Up-Converter

FFT Fast Fourier Transform

FPGA Field-Programmable Gate Array

GigE Gigabit Ethernet

IF Intermediate Frequency

LFM Linear Frequency Modulation

LO Local Oscillator

LOS Local Oscillator

MIMO Multiple-Input Multiple-Output

MO Main Oscillator

NL-LS NonLinear Least-Squares
OLS Ordinary Least-Squares
OOT Out-Of-Tree
OTA Over-The-Air
PLL Phased-Locked Loop
PMT Polymorphic Type
ppb Parts Per Billion
ppm Parts Per Million
PPS Pulse-Per-Second
PRT Pulse Repetition Time
RF Radio Frequency
RX Receiver
SAR Synthetic Aperture Radar
SDR Software-Defined Radio
SNR Signal-to-Noise Ratio
TDMA Time-Division Multiple Access
TOA Time-Of-Arrival
TOF Time-Of-Flight
TX Transmitter
UHD USRP Hardware Driver
USRP Universal Software Radio Peripheral
VCO Voltage-Controlled Oscillator
WLS Weighted Least-Squares