Evolution of a Monostatic RFSoC Platform to a Quasi- Bistatic Radar with Over-the-Air Synchronization

Alexander La
al0376la-s@student.lu.se

Matevž Stopar
ma6486st-s@student.lu.se

Department of Electrical and Information Technology
Lund University

Academic Supervisors: Liang Liu and Sijia Cheng

Company Supervisors: Anteneh Atumo Gebremariam and Henric Broström

Examiner: Erik Larsson

June 17, 2026

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

This thesis presents the evolution, design, and test of an Orthogonal Frequency Division Multiplexing (OFDM) based radar system. The application of this research question involves the implementation of an Integrated Sensing and Communication (ISAC) radar system that can be seamlessly integrated into existing wireless communication networks. The primary objective of this thesis project is to integrate RX timing synchronization between two RF frontend modules and a carrier frequency compensation scheme, enabling a quasi-bistatic setup for the radar test bench. The former iteration of the radar test bench is a monostatic setup with both the RX and TX RF frontend modules synchronized, working together as a full duplex. This work separates the two RF frontends to a distance comparable to the distance from the sensing targets and then to rely on over-the-air (OTA) synchronization methods.

Both the OTA timing synchronization and Carrier Frequency Offset (CFO) compensation schemes have been implemented on the programmable logic of a AMD Zynq UltraScale+ RFSoC ZCU208. The results and functionality of both timing synchronization and CFO compensation schemes have been validated in both RTL simulation and hardware implementation and testing. The radar detection performance of the system has proved favorable with the ability to resolve the range of reflective targets. Doppler and velocity estimation has proved more limited. This work provides a scalable foundation for a robust, hardware-accelerated ISAC radar system for future communication technologies and systems.

Keywords

OFDM, Radar, ISAC, Bistatic, Radio Frequency, PYNQ, Field Programmable Gate Array, Integrated Sensing and Communication, Carrier Frequency Offset.

Acknowledgments

Alexander

The completion of this thesis marks the end of an exciting chapter in my life and I am eternally grateful to have had the opportunity to study and work in Lund, Sweden for the past two years. The decision to uproot my life to pursue this degree was no less than one of the greatest adventures I could have undertaken. To all my friends and colleagues starting new professional lives in Sweden, I wish you all the utmost best and brightest futures imaginable.

Thank you to all my mentors directly for this project - Liang, Sijia, Anteneh, and Henric - as without your knowledge and wisdom this project would not have been possible. Your technical expertise and feedback pushed us to deliver the strongest project we could muster. Each of you, at one point or another, reached into the fog of our disorientation and guided us back towards the path of clarity.

To my thesis partner Matevž, it was truly a pleasure to work with you for the last year and a half. The daily toil and struggle was thankfully not spent alone. Best of luck with your new role at Axis Communications. Knock 'em dead.

To all my friends and connections I have made along the way, without you all, I would have surely lost myself somewhere along the road. Each and everyone one of you I have shared a laugh with, explored a new hobby, or any small adventure, you have all made an imprint on my life and I hope to have done the same.

To my family Holly, Henry, and Ashley, thank you for your never ending cheerleading and unwavering support. I will see you all very soon!

Matevž

The completion of this project represents an epilogue to one chapter, yet at the same time a prologue to another, because the people involved will no doubt have an immense effect on the continuing story, in ways that is hard to put into words.

I would like to thank my thesis partner, Alex. It was a great privilege to work with you. You were a wonderful companion throughout this process and I will miss our daily work together. I truly wish you all the best in everything that comes next.

For sharing all the knowledge, expertise and offering professional guidance, I thank our mentors Liang, Sijia, Anteneh, and Henric.

My most special thanks go to my partner Ana, for encouraging me through the hardest parts of my journey and believing in me without a single doubt. Without you, this work would not have been possible. To my family — Samo, Lea, Ema, and Neža — thank you for your endless inspiration, guidance, and love. You are my greatest role models.

Abbreviations

ADC	Analog-to-Digital Converter
ASIC	Application-Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
AXI	Advanced eXtensible Interface
AXIS	Advanced eXtensible Interface Stream
BF	Beamforming
BRAM	Block Random Access Memory
CFO	Carrier Frequency Offset
CLB	Configurable Logic Block
CORDIC	Coordinate Rotation Digital Computer
CP	Cyclic Prefix
CPO	Carrier Phase Offset
DAC	Digital-to-Analog Converter
DFT	Discrete Fourier Transform
DMA	Direct Memory Access
DSP	Digital Signal Processing
FFT	Fast Fourier Transform
FIFO	First-In, First-Out
FPGA	Field-Programmable Gate Array
FSM	Finite State Machine
GPIO	General Purpose Input/Output
IFFT	Inverse Fast Fourier Transform
ILA	Integrated Logic Analyzer
ISAC	Integrated Sensing and Communication
ISI	Inter-Symbol Interference
IP	Intellectual Property
LoS	Line of Sight

LUT	Look-Up Table
MMIO	Memory-Mapped Input/Output
OFDM	Orthogonal Frequency-Division Multiplexing
OTA	Over-the-Air
PL	Programmable Logic
PLL	Phase-Locked Loop
PS	Processing System
PYNQ	Python Productivity for Zynq
QAM	Quadrature Amplitude Modulation
RAM	Random Access Memory
RF	Radio Frequency
RFDC	RF Data Converter
RFSoC	Radio Frequency System-on-Chip
RTL	Register Transfer Level
RX	Receiver
SFO	Sampling Frequency Offset
SNR	Signal-to-Noise Ratio
SSR	Super Sample Rate
STO	Symbol Timing Offset
TX	Transmitter
URAM	Ultra RAM

Popular Science Summary

Every text message sent, every song streamed on a train journey, and every scroll through social media rely on the invisible web of wireless signals. Today, these signals have one job: carrying data. But what if this same infrastructure could be used to "see" the physical world around us?

Our research focuses on Integrated Sensing and Communications (ISAC), a technology that transforms ordinary communication signals into an accurate radar system. By harnessing the waves and signals that are already bouncing around us, we can track autonomous drones, monitor traffic flow, or coordinate robotic logistics without installing a single new sensor. This project has successfully developed a system that aims to serve three purposes. We can utilize an already existing communication system to track the distance and speed of the targets in real-time. By implementing the design on field-programmable gate array (FPGA) hardware, we can harness the high data throughput possible with a hardware-accelerated implementation of the communication and sensing system. With a hardware implementation, we can process both sensing and communication much faster than a standard computer running a script. As wireless networks utilize multiple base station sites to create a communication network, it is advantageous to use multiple devices that work in tandem with each other. Using multiple communication and sensing sites together creates distinct advantages over just using one physical location.

This project focuses on the necessary engineering changes required to turn a single site sensing system in a dual site sensing radar system. To enable such a system, synchronization is a major challenge to get multiple sensing devices to work as one cohesive system. We have targeted two important synchronization concepts that include timing synchronization between the two separated sites and a mismatch in

frequency caused by using two separated clocks. With these implemented changes we were able to verify their performance and certify that the radar capabilities of the system remained functional. With our research implemented at scale, we can move towards a future where our current wireless networks can be used as an intelligent sensor zone. A potential digital nervous system that can make autonomous transport safer and make the digital world more aware of the physical one.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Wireless communication systems are an area of technology that is constantly evolving to meet customer needs, become more technically capable, and profitable for the organizations that provide development and infrastructure for this vital technology. Research and development into Integrated Sensing and Communications (ISAC) systems provides a capable sensing system that utilizes existing or planned communication infrastructure. Although the technology candidate is not in widespread use yet, many different research organizations are developing it for integration in wireless communication systems and advanced applications. OFDM radar provides a unique opportunity to enhance an already existing OFDM-based system into one that can achieve a high level of detection capability. OFDM achieves several significant and relevant advantages as a digital communication modulation framework. OFDM provides high spectral efficiency, robustness in multipath environments, and precise range and Doppler estimation. Most importantly, OFDM has already been used in 4G and 5G standards and is expected to remain the main technology for the future of 6G wireless networks.

For practical future applications of an OFDM-based Radar sensing system, it is useful to have multiple antennas that are separated by a considerable distance and working together. The physical structure of pre-existing wireless infrastructure is a distributed network with many antennas and base station sites spread apart in many locations. This project explores expanding an already existing OFDM-based radar test bench and evolving the setup to work as a quasi-bistatic system. This will include severing some of the already existing internal synchronization and only relying on over-the-air (OTA) synchronization schemes. This encompasses

methods of timing and CFO compensation that rely on the received data. By exploiting the beamforming capabilities of the already existing monostatic system, a usable scheme is developed that is able to periodically return to a Line of Sight (LoS) boresight beam direction. This LoS beamformed position and symbol will be used as a repetitive reference for compensation and estimation of the received signal information.

The thesis project is sponsored and supported by Ericsson's Lund office and is part of their research organization. This project is a continuation of effort from two previous years of thesis project work focusing on the development of a radar test bench setup under the same research group at Ericsson. The test bench is built upon the AMD Zynq UltraScale+ RFSoC ZCU208 Evaluation Kit with two Sivers Semiconductor EVK06005 60 GHz RF Module EVK Frontends. The main aims of this work on a radar system are to extend and improve the current setup into a quasi-bistatic configuration with over-the-air timing synchronization and carrier frequency offset estimation and compensation.

1.2 Goals

1.2.1 Problem Description and Scope

This thesis project builds on two prior master's thesis projects [1]. The first project in 2024 covered the development and implementation of the underlying architecture for a usable ISAC OFDM-based radar system, which was integrated into the RFSoC FPGA and utilized lower-bandwidth RF frontends. This system was implemented as the basis for OFDM-based radar sensing and processing, but was developed under some hardware and developmental limitations. The lower bandwidth RF frontends did not allow for a favorable range and velocity resolution.

The second year of development took place in 2025 and sought to enhance the functionality and resolution of the radar test bench system. The overall resolution of the radar was increased with further gains in range and velocity resolution by implementing higher bandwidth RF frontend hardware and increasing the overall data throughput of the programmable logic [2]. This increased radar resolution was applied to different hand signal recognition algorithms, where precision in distance from the radar setup and velocity measurements is paramount to detect different hand signal positions. To take the development of the radar test bench even further, this thesis project aims to evolve the current monostatic setup into a quasi-bistatic radar setup. This configuration requires the RX frontend and the TX frontend to be separated by a distance comparable to the detectable target's range, and imposes the constraint that both carrier frequency synchronization and timing synchronization rely solely on OTA timing synchronization algorithms.

A note on the designation of this system as a quasi-bistatic configuration must be made. Although the TX and RX front ends are physically separated and operate independently, a direct internal link still exists within the FPGA hardware between the TX and RX processing blocks. This link allows the receiver to access the known transmitted signal directly, which is used as the reference in the radar range processing, specifically, the division of the received spectrum by the transmitted spectrum to extract the target response. Without this internal connection, the receiver would have no knowledge of the transmitted waveform and would instead need to reconstruct it independently through additional processing steps, such as channel estimation and equalization, re-encoding, and re-modulation, before the radar processing chain could be applied. The fully bistatic design, in which this link is eliminated entirely, is described in [3] and serves as the intended direction for future development.

1.2.2 Target Questions

- What engineering design changes are necessary to adapt an existing monostatic OFDM-based radar hardware architecture into a capable and practical quasi-bistatic setup?
- What algorithms and calculated schemes can be used in the existing system to compensate for systematic errors caused by the separation and loss of physical synchronization between TX and RX communication?
- Can we explore different alternative detection and synchronization schemes using the already built-in beamforming capabilities of the current radar system?

1.3 Project Developement Process

- Literature Review: Search and digest available material through papers, articles, and textbooks on the background of the technical details required to understand the current project and future improvements. An enormous amount of information on wireless systems, OFDM signals, radar, and synchronization algorithms had to be reviewed in order to build a strong foundation of knowledge to successfully progress throughout the project. Another important component of the literature review cycle was to familiarize ourselves with the past years' documentation of the project and to conduct short interviews with one of the past project members about the operational organization of the Vivado FPGA project.
- Python Model: A Python-based model was created to base a foundational understanding of OFDM communications systems and applications of OFDM

with radar systems. This Python model explores and simulates many foundational topics necessary to understand the functional design of the larger radar system architecture. The main features covered are OFDM data generation, OFDM modulation and demodulation, receive and transmit systematic error estimation and compensation, and radar Range-Doppler processing. This environment was extremely useful for capturing reference data against a digital design system, generating input data, and testing both a chosen timing algorithm and frequency compensation method.

- Create and implement onto the hardware novel IP blocks: Each additional feature of the existing radar test bench contributes to a comprehensive bistatic radar system architecture. The location in the architecture and the RF data chain is outlined in 4.7
  - Minn Method-based timing algorithm: The Minn Method frame timing algorithm is a method for timing synchronization to obtain knowledge of where each OFDM signal frame starts. In a real system, the receiver does not know at which point in time each OFDM symbol or OFDM frame starts and ends. The most well-known method for OFDM timing synchronization is known as the Schimdl and Cox OFDM technique [4]. A known flaw for the Schmidl–Cox is a lack of precision due to the method's reliance on a calculated metric used for synchronization with a flat plateau that manifests as temporal ambiguity. To resolve this known issue, a technique that this project will refer to as the Minn Timing Method was implemented in hardware.
  - TX BRAM Reader Reorganization: The TX BRAM Reader needed output reorganization to support a chosen OFDM data frame structure. The frame structure consists of a single preamble symbol along with a repeated structure of repeated LoS and data symbols to create each transmitted frame.
  - LoS Symbol Capture: The LoS symbol capture is the chosen mechanism to store each LoS symbol from each frame "chunk" for use in the RX divider for radar processing.
  - Divider Control Block: This novel IP block is used to orchestrate the OFDM symbol-based division operation used for radar processing with a chosen LoS symbols from the TX data path and compare those against the incoming data symbols from the RX data path.
  - CFO Estimation and Compensation: CFO estimation and compensation addresses the frequency mismatch that arises from the separation of the transmitter and receiver local oscillators. Without the correction, the CFO will introduce a time-varying phase rotation across OFDM subcarriers. The unwanted phase rotation will degrade communication and radar sensing performance. The CFO estimation functional block leverages the already existing calculated metric for tim-

ing synchronization from the inserted preamble symbol and converts it into a measurable phase angle. The compensation stage applies a corresponding counter rotation in the time domain prior to the Fast Fourier Transformation demodulation.

  - Novel Beamforming Indexing Pattern: Beamforming functionality is already implemented into the radar design. A return function instruction had to be re-wired to be used by the TRX BFO1 chip on the RF Frontends for a unique beam forming indexing pattern.

- Integrate novel IP blocks into the existing OFDM radar system design: Following individual development, testing, and verification of each novel IP block, the blocks were integrated into the new OFDM radar system design. Each block was connected within the established RF data chain, with attention paid to interface compatibility, data flow, and timing across both RX receive and TX transmit domains. Integration testing was conducted incrementally to isolate and resolve any underlying IP block design faults or implementation bugs.
- Performance Evaluation and Testing: System performance was evaluated through a combination of testing and analysis of captured RF data and Integrated Logic Analyzer (ILA) waveforms. Captured data was analyzed to ensure that timing synchronization and CFO estimation and compensation functional blocks were working within acceptable operating ranges. Field testing of the radar performance was completed in a low interference indoor environment to categorize the system with minimal interferences. The subsequent range-Doppler plots were analyzed against physical target measurements to ensure target range accuracy.

1.4 Report Organization

- **Chapter 1: Introduction** establishes the project's context and background, defines the core problems, and outlines research methods and report organization.
- **Chapter 2: Technical Overview** shall present the theoretical and technical foundation in order to understand the system design, operation, and applications of the presented ISAC OFDM hardware implemented radar system. The chapter explains key concepts of OFDM modulated data, OFDM data structure, radar processing, radar beamforming, and the systematic offsets that systems are designed to compensate for and maintain signal integrity.
- **Chapter 3: Hardware System Design and Software Implementation** details the novel hardware designs implemented as Intellectual Property (IP) within the Programmable Logic (PL), as well as the software

architecture within the Processing System (PS) that manages the FPGA hardware.

- **Chapter 4: Methodology** describes the methods, processes, and platforms used to complete the project. This includes the hardware and software platforms utilized to build the project and their utility within the larger system and experimental setup.
- **Chapter 5: Results and Analysis** provides overall results evaluation incorporating analysis of the final system results and performance. This section also examines the quasi-bistatic development features and their utility to the overall system.
- **Chapter 6: Conclusions and Future Work** will summarize key findings, contributions, and closing thoughts on the results of the proposed and implemented OFDM ISAC radar system. The report concludes by addressing the limitations and potential future work for the further development of this radar test bench system.

Chapter 2

Technical Overview

2.1 OFDM Fundamentals

2.1.1 OFDM Basics

As established in [5], in a typical single carrier system, a sequence of symbols $\{a_n\}$, where n denotes the symbol index, is transmitted at a rate of one symbol every T seconds. After shaping by a transmit filter, propagation through the channel, and processing by the receive filter (and possibly an equaliser), the time continuous signal at the receiver can be written as:

$$y(t) = \sum_{m=-\infty}^{\infty} a_m g(t - mT) + z(t) \tag{2.1}$$

where $g(t)$ denotes the impulse response of the overall end-to-end system, and $z(t)$ is the additive noise.

For simplicity, assume that the effect of the channel is perfectly compensated. Therefore, the overall impulse response is subject to transmit and receive filter only. When the noise term is ignored, the sampling at $t_n = nT$ gives:

$$y(t_n) = \sum_{m=-\infty}^{\infty} a_m g((n - m)T) \tag{2.2}$$

When isolating the n-th sample to detect a_n, the equation can be rewritten in the

following form:

$$y(t_n) = a_n g(0) + \sum_{m=-\infty, m\neq n}^{\infty} a_m g((n-m)T) \tag{2.3}$$

The first term in (2.3) represents the desired symbol scaled by the pulse amplitude at the sampling instant, while the second term aggregates the contributions of all other transmitted symbols evaluated at the same sampling point. Since $g(t)$ cannot be strictly time-limited, a consequence of the finite bandwidth of any practical channel, the pulse response does not vanish exactly at the neighboring sampling instants. Whenever $g((n-m)T) \neq 0$ for $\forall m \neq n$ the second term is non-zero, and the detection of a_n is corrupted by contributions from surrounding symbols. This phenomenon is known as inter-symbol interference (ISI). As the data rate increases, the symbol period T decreases, compressing the time available for the pulse to decay between consecutive sampling instants and thereby making ISI progressively more severe. To eliminate ISI entirely, the pulse shape $g(t)$ must satisfy the Nyquist criterion:

$$\sum_{i=-\infty}^{\infty} G(f - \frac{i}{T}) = T \tag{2.4}$$

where $G(f)$ is the Fourier transform of $g(t)$. Equivalently, $g(kT) = 0$ for all integer $k \neq 0$. Following the Nyquist criterion, ISI can be eliminated even for short symbol periods. In order to support the symbol rate $R_s = 1/T$, the minimum Nyquist bandwidth is $R_s/2$.

Perfect equalization can break down in wireless channels with multipath propagation. When the signal bandwidth exceeds the channel coherence bandwidth, the channel is frequency selective. Different frequency components experience different fading characteristics and produce ISI that is harder to equalize. This motivates multi-carrier transmission: by dividing the signal into many narrower subcarriers (each narrower than to the coherence bandwidth), each subcarrier sees approximately flat fading, and ISI is reduced or more easily equalized.

One of the more popular multi carrier transmission schemes is OFDM. OFDM transmits many orthogonal subcarriers that overlap in frequency but remain orthogonal because the subcarrier spacing is $1/T$ (the symbol rate). This can be viewed as a generalization of the single-carrier Nyquist condition 2.4: the subcarrier pulse spectra are shifted copies that together satisfy the Nyquist sum condition. In practice, the discrete Fourier transform and inverse discrete Fourier

transform are often used for implementing these orthogonal signals, since both can be implemented efficiently using the Fast Fourier transform and Inverse Fast Fourier transform, respectively.

A high data rate is desirable for many applications of wireless communication technologies. The growing complexity of ISAC systems for practical use not only demands high data throughput but also accurate and simultaneous environmental detection. ISAC systems can address both requirements within a unified system by exploiting a single shared waveform for simultaneous radar sensing and data transmission. The multi carrier orthogonal structure of OFDM lends itself to the wide band coherent structure required for range-Doppler radar processing.

OFDM is a multi carrier modulation technique that transmits data simultaneously over a multiple orthogonal subcarrier structure. The concept was first proposed in the 1960s [6] and later refined with the introduction of the discrete Fourier transform as an efficient implementation mechanism [7]. OFDM has emerged as the transmission scheme of choice, having already been adopted in standards such as LTE, 4G, and 5G [8] [9]. The same structural properties that make OFDM effective for communication, also make it a powerful radar waveform.

The subcarriers are spaced such that orthogonality is preserved across the symbol duration T. The k-th subcarrier frequency is:

$$f_k = f_0 + k\Delta f, \qquad k = 0, 1, \ldots, K-1 \tag{2.5}$$

The subcarrier spacing between each subcarrier can be described as:

$$\Delta f = \frac{1}{T_{sym}} \tag{2.6}$$

where T_{sym} is the useful OFDM symbol duration, excluding the cyclic prefix. This spacing is the key to the orthogonality property of OFDM. Each subcarrier takes the form of a sine cardinal (sinc) function in the frequency domain, and the spacing of $\Delta f = 1/T_{sym}$ ensures that the peak of every subcarrier coincides precisely with the zero crossings of all other subcarriers:

$$\mathrm{sinc}(x) = \frac{\sin(\pi x)}{\pi x} \tag{2.7}$$

This mutual alignment means that even through the adjacent subcarriers overlap

significantly in the frequency domain, they do not interfere with one another when sampled at their respective center frequencies. This quality allows OFDM waveforms to have tightly packed subcarriers while maintaining perfect orthogonality under ideal channel conditions.

The transmitted complex symbols $c_{k,l}$ are selected from a chosen *modulation alphabet*. Standard modulation alphabets include BPSK, QPSK, and higher-order QAM constellations [10]. Each symbol $c_{k,l}$ modulates associated subcarriers k during the OFDM symbol period l. This encodes information as a complex amplitude and phase value represented by the location of the chosen constellation position point.

2.1.2 Cyclic Prefix

OFDM transmissions commonly use a "guard interval" between transmitted symbols to protect against ISI. Two common options are through zero-padding between OFDM symbols and the cyclic prefix (CP); the latter is used in this work. Let x define the time domain OFDM symbol

$$x = [x_0, x_1, \ldots, x_{N-1}]^T \tag{2.8}$$

The cyclic prefix is the insertion of a set amount of symbol data from the end of the OFDM symbol and the prepending of that data to the start of the symbol. The CP of length L prepends the last L samples to the front.

$$x_{cp} = [x_{N-L}, \ldots, x_{N-1}, x_0, x_1, \ldots, x_{N-1}]^T \tag{2.9}$$

The primary advantage of cyclic prefix over zero padding is that it preserves circular convolution in the frequency domain. Due to the properties of the Discrete Time Fourier Transform (DTFT), a circular convolution in the time domain becomes a simple multiplication in the frequency domain [11]. The length of the cyclic prefix must be greater than the channel delay to maintain orthogonality among the subcarriers.

$$T_{total} = T_{sym} + T_{CP} \tag{2.10}$$

Within the design and the RF data chain, IP blocks are implemented for both cyclic prefix add and cyclic prefix removal in the Transmit block and Receive block, respectively. The CP add and CP remove operations are only done in the time domain, following the IFFT in the Transmit block and before the FFT in

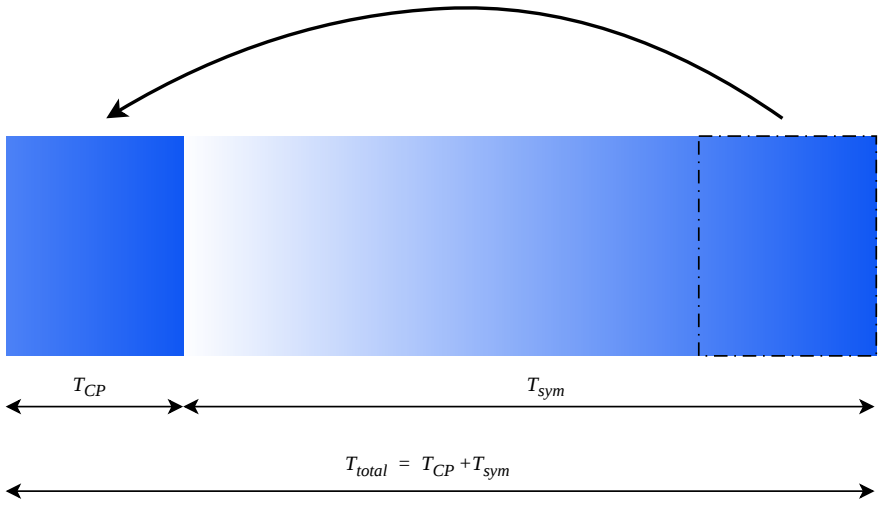

Figure 2.1: Cyclic prefix prepend duplication.

the Receive block. The additional length added by the cyclic prefix must be taken into account when transmitting and receiving OFDM data.

2.1.3 OFDM Modulation and Demodulation

The main goal of OFDM modulation is to split a high-speed data stream into many slower parallel streams. Each stream is transmitted over a closely spaced, overlapping, subcarrier frequencies. This allows the system to be highly efficient as the data bits are distributed across multiple subcarriers and transmitted simultaneously. As each individual subcarrier transmits data at a much lower rate than the original high speed stream, the symbol duration becomes long relative to the channel delay spread, making the signal highly resistant to multipath fading. OFDM modulation is efficiently implemented in the digital domain using the Inverse Fast Fourier Transform (IFFT). For a given OFDM symbol l, the K active frequency-domain symbols $c_{k,l}$ are mapped onto K of the N available IFFT bins. The remaining $N - K$ bins are set to zero to realize guard and null subcarriers. The IFFT then converts this frequency domain representation into a time domain signal of length N according to [12]:

$$x_l[n] = \frac{1}{N} \sum_{k=0}^{N-1} c_{k,l}\, e^{j2\pi kn/N}, \qquad n = 0, 1, \ldots, N-1 \tag{2.11}$$

where $x_l[n]$ is the n-th time domain sample of the lth OFDM symbol. $c_{k,l} = 0$ for all inactive subcarrier indices k. The IFFT operation inherently preserves

subcarrier orthogonality, which allows all K active subcarriers to occupy the same total bandwidth simultaneously without interfering with one another. Following the IFFT, a cyclic prefix is prepended to each symbol prior to transmission, as described in Section 2.1.3

At the receiver, after CP removal, the time-domain signal is converted back to the frequency domain by applying the Fast Fourier Transform (FFT) of length N as follows [13]:

$$\hat{c}_{k,l} = \sum_{n=0}^{N-1} x_l[n]\, e^{-j2\pi kn/N}, \qquad k = 0, 1, \ldots, N-1 \tag{2.12}$$

For a distortionless channel, this recovers the original frequency-domain symbols at all N bins, of which the K active subcarrier outputs carry the transmitted data symbols. In the presence of a multipath channel, the received time-domain samples are the circular convolution of the transmitted signal with the channel impulse response $h[n]$. As established in Section 2.1.2, the cyclic prefix converts linear convolution into circular convolution, allowing the channel effect to be expressed as a simple element-wise multiplication in the frequency domain for each active subcarrier $k = 0, 1, \ldots, K-1$:

$$\hat{c}_{k,l} = c_{k,l} \cdot H[k] + w_{k,l}, \qquad k = 0, 1, \ldots, K-1 \tag{2.13}$$

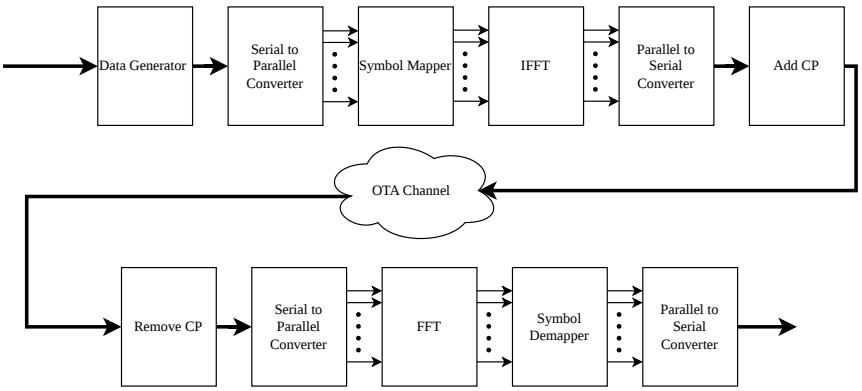

Figure 2.2: Generic OFDM modulation and demodulation chain.

2.1.4 Generic OFDM Frame Structure

Transmitted OFDM data is organized into frames, each consisting of M OFDM symbols in the time domain and K subcarriers in the frequency domain. The accumulation of these complex symbols $c_{k,l}$ across all the active subcarriers K and symbols M within a frame is represented as the matrix below [1].

$$\mathbf{F_{TX}} = \begin{bmatrix} c_{0,0} & c_{0,1} & \cdots & c_{0,M-1} \\ c_{1,0} & c_{1,1} & \cdots & c_{1,M-1} \\ \vdots & \vdots & \ddots & \vdots \\ c_{K-1,0} & c_{K-1,1} & \cdots & c_{K-1,M-1} \end{bmatrix} \tag{2.14}$$

This yields a grid of $K \times M$ complex-valued resource elements. Not all of the symbols or resource elements carry data, but instead the frame can be partitioned into distinct sections that serve different functional roles. As established in [14], *Preamble* that occupies the leading OFDM symbols can be used for timing synchronization purposes. *Pilot* subcarriers can be interspersed in different locations with the resource grid for channel tracking and compensating channel effects. Finally, a *Guard Interval* (such as a cyclic prefix) can be inserted between symbols to alleviate the effects of ISI, while *Guard Subcarriers* are placed at the edges of the grid to prevent adjacent channel interference.

2.1.5 Super Sample Architecture

When high sampling rates are required, processing one sample per clock cycle would demand clock frequencies beyond the practical limits of the hardware. To overcome this problem, systems may employ a Super Sample Rate (SSR) architecture, in which multiple samples are processed in parallel within each clock cycle. This effectively decouples the processing clock frequency from the sample rate, enabling real-time high-bandwidth operation within the timing constraints of the hardware.

2.2 OFDM Radar Processing

2.2.1 Monostatic Radar Setup

A monostatic radar setup is categorized as a radar system where the TX unit and the RX unit are physically located in the same location and share structural and hardware resources [15]. Monostatic radar setups can even use the same set of antennas both for transmit and receive as a duplex system. Monostatic radar systems employ the same oscillator for TX and RX to keep the two signals coherent with each other.

The past iteration of the project was developed as a fully monostatic system with both RX and TX frontend modules facing the target in parallel and very close proximity to each other. The central challenge of this work was to identify a sound and efficient method to decouple the system into a bistatic configuration. Special attention had to be paid to how the synchronization of the system would work when both TX and RX frontends become decoupled, and the various systemic error compensations to be made to the system to receive a coherent range-Doppler map.

2.2.2 Bistatic Radar Setup

A bistatic radar setup is categorized as a radar system with the TX transmit unit and the RX receive unit separated into two geographically or spatially distinct sites [15]. The target object of detection is sensed via the reflected energy from the transmitter off the object and into the receiver. Special consideration has to be given for the additional "time of flight" for the reflected signals and the lack of inherent synchronization between the TX and RX. There exists a third radar configuration called a multistatic radar setup that is classified as an extension of bistatic radar in which multiple transmit and multiple receive units are used as a network of nodes to detect a target [15].

There are a number of engineering challenges that arise when converting a monostatic radar system to a bistatic radar system.

- Free-running individual clocks on each frontend in comparison to the perfectly synchronized and matched clocks of the monostatic setup
- The transmit delays between the transmit and receive module due to the physically separated transceiver units.
- The loss of timing synchronization
- The necessity to physically separate the RF frontend modules when testing
- Altering the beamforming indexing pattern to accommodate a physical configuration where the TX and RX frontends are separate and facing each other

Critical design changes must be constructed and incorporated into the monostatic radar system to overcome the systematic errors that arise when designing a bistatic system. The chosen modifications to the existing radar project are intended to address these engineering challenges without impacting the already existing system.

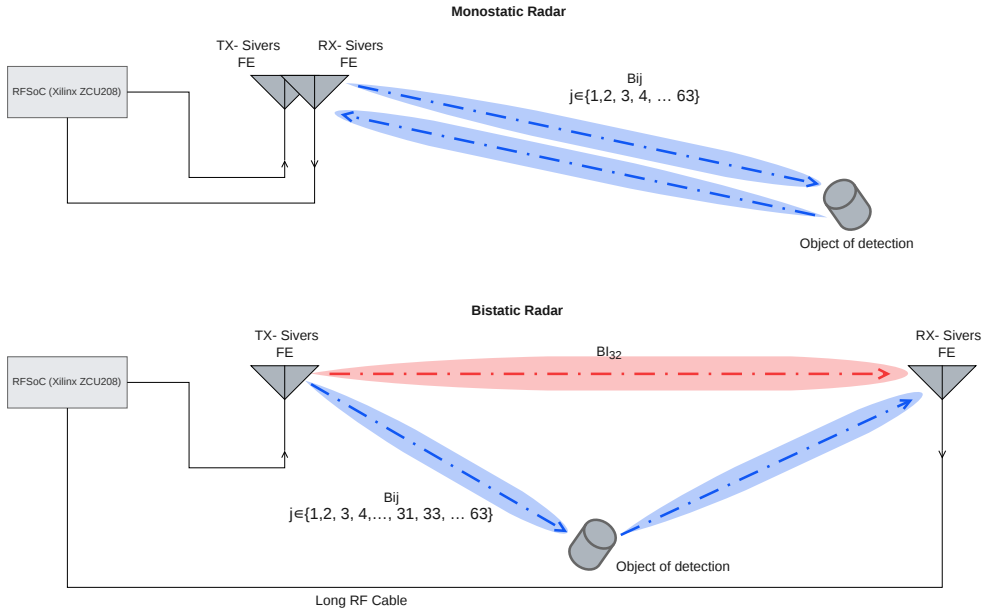

Figure 2.3: Monostatic and bistatic setup comparison.

Throughout this work, the term LoS symbol is used frequently, and therefore it is important to establish a clear understanding of the concept. Most data symbols are used for sensing, which means that different beam directions are used to scan the surrounding environment. When these symbols reach the receiver via reflections from targets in the scene, the OFDM signal is affected by the target response, which enables the core radar functionality. However, for a small subset of symbols, this effect is undesirable, as these symbols are not intended for sensing. Instead, they serve either for the timing synchronization or as reference symbols against which the other sensing symbols are compared in order to extract range and Doppler information. To minimize the influence of the surrounding environment on these symbols, they are transmitted directly on the boresight beam, meaning they travel via the LoS path between transmitter and receiver. The symbol used for timing synchronization will be referred to throughout this work as the preamble. The reference symbol transmitted on the boresight beam, by contrast, has no established terminology and will therefore be referred to as the LoS symbol.

2.2.3 Quasi-Bistatic Radar Setup Specification

In a bistatic OFDM radar system, the receiver needs to know exactly what OFDM symbols are transmitted to divide out the received symbols to extract range and Doppler information. In the monostatic radar system this is trivial, as the trans-

mitter and receiver have a direct connection to each other. All symbols are sent directly from the transmit block to the receive block with the exact same clock signal for direct comparison. In the bistatic configuration, acquiring the reference signal at the receiver is a critical challenge that requires precise synchronization schemes.

In this work, a clear distinction between the fully bistatic radar setup and the quasi-bistatic radar setup was established. As established in [3], a fully realized bistatic OFDM radar setup requires a transmit frame recovery data path for full frame radar signal processing. This recovery path comprises channel estimation, equalization, demodulation, re-encoding, and re-modulation of the received symbols. After additional processing element-wise division with the reconstructed reference can be performed to complete the radar processing chain. To incorporate all these features onto hardware in the context of this work poses two hurdles. The amount of work would exceed the scope for a master's thesis work, and functionality would need to be distributed across two FPGA boards due to utilization limitations. A compromise in the design was reached, where a novel design was formulated for the quasi-bistatic configuration.

The quasi-bistatic framing of the proposed system is well-suited to evaluate the fundamental synchronization, timing, and channel offset features required for a fully bistatic configuration, within the constraints of a resource-limited single-board implementation. The reference signal is deliberately orchestrated by the hardware design, and the frontend devices remain physically separated with no shared clock. The proposed system retains the key characteristics of a bistatic setup, including OTA synchronization, systematic channel offset handling, and a mechanism to perform radar processing against a set of reference signals. In the proposed architecture, radar processing is performed on the captured LoS symbols in place of fully-recovered transmitted symbols. In the technique shown in [3], the reference quality is bounded by the decoder performance. In the system introduced in this thesis, the reference quality depends on how accurately the LoS symbol is timed and captured. Although this design moves toward a fully bistatic radar system, the limitation remains that a direct connection exists between the TX and RX processing blocks. Future work would include compensation and estimation schemes for Symbol Frequency Offset (SFO) and Carrier Phase Offset (CPO) and an enhanced data chain to complete full transmit frame recovery.

Table 2.1 summarizes the key architectural design differences between a fully bistatic OFDM radar system, as described in [3], and the quasi-bistatic configuration implemented in this thesis project.

Table 2.1: Comparison of Fully Bistatic and Quasi-Bistatic OFDM Radar Configurations.

Feature	**Fully Bistatic**	**Quasi-Bistatic (This Work)**
TX/RX clock source	Independent free-running oscillators	Independent free-running oscillators
Physical separation	Physically separated, no shared clock	Physically separated, no shared clock
Reference signal acquisition	Full frame recovery via channel estimation, equalization, demodulation, re-encoding, and re-modulation	LoS symbol capture and direct forwarding to radar processing chain
TX-RX connection	No direct connection	Direct connection retained for LoS reference transfer
OTA synchronization	Implemented using Schmidl-Cox algorithm	Implemented with Minn [16]
Clock offset compensation	Required (SFO, CFO, CPO estimation)	Partial – CFO compensation implemented
FPGA resource utilization	Resource utilization spread across two boards	Reduced processing chain fits on a single board

2.2.4 OFDM Radar Signal Processing

Consider the matrix $\mathbf{F_{TX}}$ of size $K \times M$ that contains the transmitted complex symbols $c_{k,l}$: in equation 2.14 of section 2.1.4 where each column contains the K subcarrier symbols of the l-th OFDM symbol, and each row contains the symbol on subcarrier k across all M consecutive symbols. For a single target at range R with velocity v, the target produces a time delay of τ and a Doppler frequency shift of f_d.

$$\tau = \frac{2R}{c}, \qquad f_d^{radar} = \frac{2vf_c}{c} \tag{2.15}$$

Assume c is the speed of light and f_c is the carrier frequency. After CP removal and FFT demodulation, the received symbol on the subcarrier k of OFDM symbol l is:

$$r_{k,l} = c_{k,l} \cdot \alpha \cdot e^{-j2\pi k\Delta f\tau} \cdot e^{j2\pi f_d l T_{\text{sym}}} + w_{k,l} \tag{2.16}$$

where α is the complex target reflection coefficient, $w_{k,l}$ is the additive noise, $k = 0, 1, \ldots, K-1$ is the subcarrier index, and $l = 0, 1, \ldots, M-1$ is the OFDM symbol index.These received symbols form the received matrix $\mathbf{F_{RX}}$ of identical dimensions to $\mathbf{F_{TX}}$. Element-wise division of $\mathbf{F_{RX}}$ by $\mathbf{F_{TX}}$ removes the known transmitted symbols, yielding the radar channel matrix $\mathbf{D}$:

$$D_{k,l} = \frac{r_{k,l}}{c_{k,l}} = \alpha \cdot e^{-j2\pi k\Delta f\tau} \cdot e^{j2\pi f_d l T_{sym}} + \tilde{w}_{k,l} \tag{2.17}$$

where $\tilde{w}_{k,l} = w_{k,l}/c_{k,l}$ is a scaled noise term. This element-wise operation is at the heart of this OFDM radar system calculation and it enables the system to produce range-Doppler maps.

A 2D IFFT-FFT operational sequence is performed on the radar channel matrix to obtain range and Doppler information. Following the framework introduced by Sturm et al [17] [18], the matrix $\mathbf{D}$ encodes the target's range and velocity as separate phase progressions along two axes. These include a linear phase ramp across the subcarriers k encoding the time delay τ, and a linear phase ramp across the OFDM symbols l encoding the Doppler shift f_d (thus the velocity).

To extract both measurable quantities, a 2D IFFT-FFT is applied to the radar matrix $\mathbf{D}$. The IFFT is performed on the subcarriers and an FFT is performed on the symbols, in that order. This is a departure from Sturm's original formulation. This design will differ as IFFT is applied over the subcarriers and a subsequent FFT on the symbols compared to an FFT on the symbols and a further FFT on the subcarriers. Since the receiver has already applied an OFDM demodulation FFT after CP removal and signal recovery, the time domain signal has been effectively transformed once. Applying an additional FFT over the subcarriers would result in a double transformation and corrupt the range estimate.

2.3 Synchronization

2.3.1 Carrier Frequency Offset

In any wireless communication system, the transmitter and receiver rely on a local oscillator to perform carrier modulation and demodulation. The transmitter uses its oscillator to up-convert the baseband signal to the passband, and the receiver uses its own oscillator to down-convert the received passband signal back to baseband. Ideally, both oscillators would generate exactly the same carrier frequency. In practice, however, this is never truly the case. Real-world oscillators suffer from manufacturing tolerances, temperature sensitivities, and effects of age-related degradation. As a result, the transmitter and receiver are never guaranteed to operate at precisely the same frequency. In general, there are two types of distortion associated with the carrier signal. One is phase noise due to the instability of the oscillator hardware, and the other is caused by the Doppler frequency shift f_d. The combined result of these two effects is a Carrier Frequency Offset (CFO), a small but consequential difference between the carrier frequency the receiver expects and the one it actually receives [19].

In a single-carrier system, CFO causes a slow, manageable phase drift that can be tracked relatively easily. In an OFDM system, however, CFO is far more damaging to system performance. In the OFDM scheme, orthogonality between subcarriers is what allows each one to be demodulated independently without mutual interference. A CFO shifts all received subcarriers away from their nominal frequency positions, so they are no longer aligned with the receiver's fixed FFT demodulation basis. Since the receiver always demodulates at the nominal subcarrier frequencies, this misalignment breaks the orthogonality between the received subcarriers and the demodulation basis, causing each subcarrier to leak energy into all other FFT bins simultaneously — an effect known as inter-carrier interference (ICI). As shown by [20], even a CFO as small as a few percent of the subcarrier spacing can cause significant degradation in bit error rate, making CFO one of the most critical impairments to address in any OFDM receiver system.

If f_c and f_c' denote the carrier frequencies at the transmitter and receiver, respectively, the total frequency offset is:

$$f_{offset} = f_c - f_c' \tag{2.18}$$

The Doppler component of this offset is determined by the carrier frequency and the velocity v of the moving terminal:

$$f_d^{CFO} = \frac{v \cdot f_c}{c} \tag{2.19}$$

where c is the speed of light. Note that (2.19) slightly differs from the Doppler equation used in the previous section. Here, a standard OFDM system equation is used, whereas in the radar context, the round-trip factor is applied. It is standard practice in OFDM analysis to express the CFO as a normalized quantity relative to the subcarrier spacing Δf:

$$\varepsilon = \frac{f_{offset}}{\Delta f} \tag{2.20}$$

The second component comprises the inherent oscillator mismatch:

$$f_{osc} = f_c \cdot \delta_{osc} \tag{2.21}$$

The total CFO is therefore the sum of the Doppler and oscillator contributions:

$$f_{offset} = f_d^{CFO} + f_{osc} = \frac{v \cdot f_c}{c} + \delta_{osc} \cdot f_c \tag{2.22}$$

To illustrate the practical magnitude of CFO, consider an OFDM system, which operates at a bandwidth of 1966.08 MHz, derived from a DAC (Digital to Analog Converter) PLL (Phase-Locked Loop) output rate of 3932.16 Mega Samples Per Second (MSPS) with an interpolation factor of two, and a carrier frequency of $f_c = 60\,\text{GHz}$. This gives a subcarrier spacing of $\Delta f = f_s/K \approx 960\,\text{kHz}$.

A key parameter in CFO estimation is the accuracy of the oscillator frequency. For illustration purposes, a reference value is drawn from 3GPP documentation [21], which specifies a frequency error requirement of ± 0.05 ppm for wide-area LTE base stations, confirming that sub-ppm oscillator accuracy is required in practical radio hardware. A slightly more conservative value of 1 ppm is therefore adopted in this analysis, which remains consistent with the order of magnitude of frequency errors encountered in real radio systems.

For a pedestrian moving at $v = 3\ km/h$, the two contributions to CFO can be calculated separately. Parameters were chosen so they match the hardware system.

$$f_d = \frac{v}{c} \cdot f_c = \frac{3000/3600}{3 \times 10^8} \times 60 \times 10^9 \approx 167 \text{ Hz} \tag{2.23}$$

The oscillator mismatch component is:

$$f_{osc} = f_c \times 1 \times 10^{-6} = 60 \times 10^9 \times 10^{-6} = 60 \text{ kHz} \tag{2.24}$$

The total CFO is therefore:

$$f_{total} = 167 + 60{,}000 \approx 60.2 \text{ kHz} \tag{2.25}$$

And the normalized total CFO is:

$$\varepsilon = \frac{60{,}167}{960{,}000} \approx 0.063 \tag{2.26}$$

This result shows that even for a slowly moving pedestrian with a reasonably good oscillator, the total CFO reaches approximately 6.3 % of one subcarrier spacing. The oscillator mismatch is the dominant contributor by more than two orders of magnitude compared to Doppler at this speed.

2.3.2 OFDM Timing Synchronization Algorithms

Both synchronization methods presented in this chapter exploit an OFDM symbol, referred to as the preamble, which is attached to the transmitted frame. By constructing the preamble with a specific repeating structure, the receiver can identify the frame boundary through autocorrelation of the received signal, without requiring knowledge of the transmitted data. Section 4 will go into more detail why these specific methods were chosen.

2.3.2.1 Schmidl–Cox Algorithm

The Schmidl–Cox algorithm [4] is based on a preamble consisting of two identical halves, prepended by a cyclic prefix, as illustrated in Fig. 2.4. This property will be preserved after propagation through a linear channel.

The algorithm exploits this repetition by computing the autocorrelation between the two halves of the received signal. Each sample from the first half is conjugate-multiplied by the corresponding sample from the second half. A CFO f_{CFO} in-

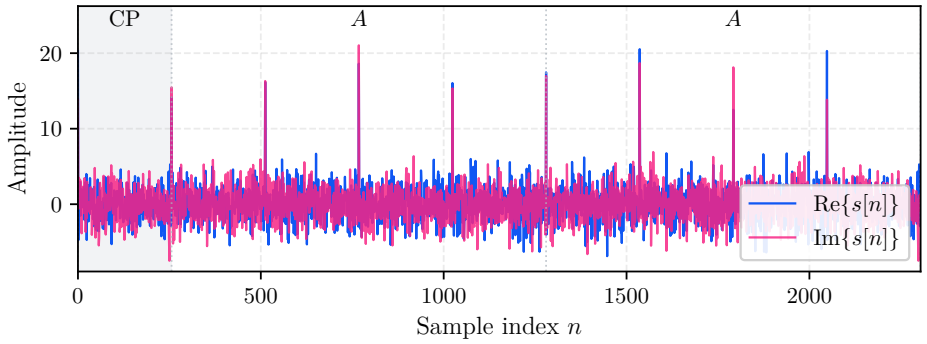

Figure 2.4: Schmidl–Cox preamble structure.

troduces a linearly growing phase rotation $e^{j2\pi \frac{f_{CFO}}{f_s} n}$ on every received sample n, where f_s is the sampling frequency. However, because the two halves are identical, this linearly growing phase cancels in each conjugate product, leaving only a residual constant phase shift $e^{j2\pi \frac{f_{CFO}}{f_s} L}$ that is the same for every pair. The resulting accumulated phase across all L pairs is therefore a direct function of the CFO, enabling its estimation. The correlation metric is defined as:

$$P(d) = \sum_{m=0}^{L-1} r_{d+m}^{*} \, r_{d+m+L} \tag{2.27}$$

where L is the number of complex samples in one half of the training symbol, excluding the cyclic prefix, and d is the sample index of the first sample in the current window of $2L$ samples. Importantly, metric $P(d)$ can be updated efficiently using the sliding-window recursion:

$$P(d+1) = P(d) + (r_{d+L}^{*} r_{d+2L}) - (r_{d}^{*} r_{d+L}) \tag{2.28}$$

This reduces each update to a single addition and subtraction, making the metric suited for a hardware implementation. To ensure robustness against variations in received signal power, $P(d)$ is normalized by the energy of the second half-symbol:

$$R(d) = \sum_{m=0}^{L-1} |r_{d+m+L}|^2 \tag{2.29}$$

It should be noted that, similar to the equation 2.28, R can also be calculated

using the sliding window method.

Using (2.27) and (2.29) a time metric can be calculated:

$$M(d) = \frac{|P(d)|^2}{(R(d))^2} \tag{2.30}$$

Before continuing on to the next method it is important to address the general shape of the metric. Understanding how a general metric shape corresponds to the frame structure will give the reader a better understanding of the design choices made in the following section. This relation is shown in Fig. 2.5 Section 4 will go into details of the calculation of the metric and the estimation of the timing.

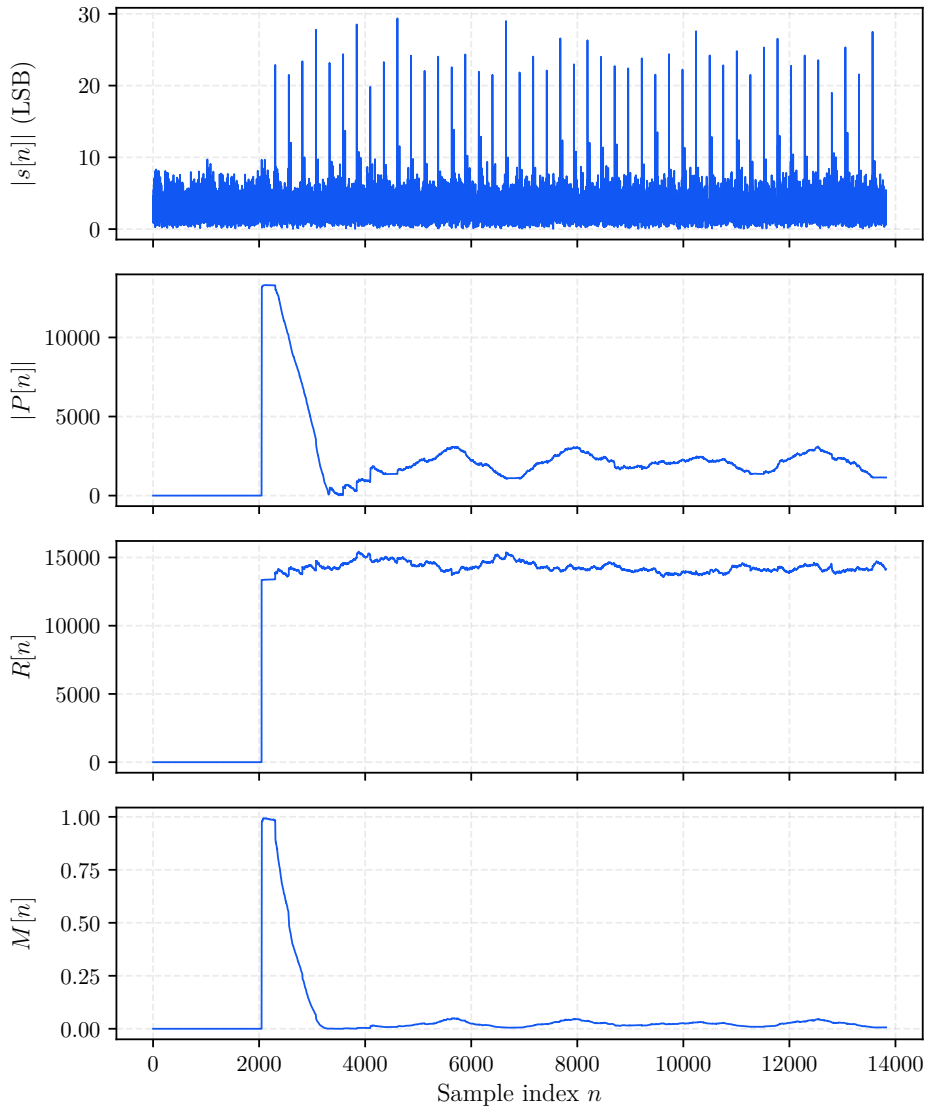

Figure 2.5: Schmidl–Cox metrics with the corresponding RX signal.

2.3.2.2 Minn Algorithm

The Minn method [16] builds upon the Schmidl–Cox approach by addressing its most prominent limitation: the plateau in the timing metric $M(d)$, visible in Fig. 2.5, which introduces ambiguity in the timing estimate. The Minn method addresses this by slightly restructuring the training symbol and the calculation of $P(d)$ is redefined to match the new sign pattern.

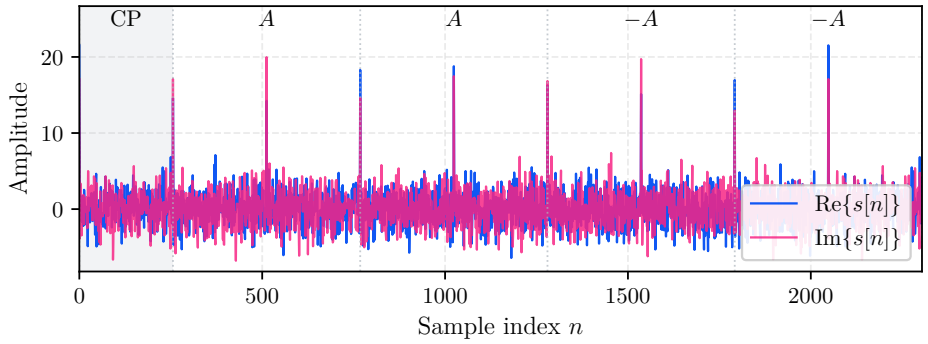

Figure 2.6: Minn preamble structure.

It is worth noting a difference in notation between the two methods. In the Schmidl–Cox method, L denotes half of the symbol length, i.e., $L = N/2$. In the Minn method, the training symbol is divided into four blocks, so from this point on L refers to the quarter of the symbol length:

$$L = \frac{N}{4} \tag{2.31}$$

Instead of two identical halves, the Minn method uses a training symbol composed of four blocks, as shown in Fig. 2.6, following the sign pattern:

$$\mathbf{s} = [\, A,\ A,\ -A,\ -A \,] \tag{2.32}$$

where A is a block of L samples generated by taking the L-point IFFT of L length modulated data from a pseudo noise sequence. The first two blocks are identical, while the last two blocks are the negation of the first two. The training symbol can therefore be written as:

$$s_n = \begin{cases} c_n, & 0 \le n < 2L \\ -c_{n-2L}, & 2L \le n < 4L \end{cases} \tag{2.33}$$

The correlation metric $P(d)$ is redefined to account for the four-block structure. Rather than correlating a single pair of half-symbols, the metric now sums the conjugate products over two separate block pairs:

$$P(d) = \sum_{k=0}^{1} \sum_{m=0}^{L-1} r^*(d + 2Lk + m) \cdot r(d + 2Lk + m + L) \tag{2.34}$$

Similarly to the Schmidl–Cox method, (2.34) can be rewritten using a sliding window approach. The splitting $P(d) = P_1(d) + P_2(d)$, where P_1 covers the first block pair and P_2 covers the second:

$$P_1(d+1) = P_1(d) + r^*_{d+L}\, r_{d+2L} - r^*_d\, r_{d+L} \tag{2.35}$$

$$P_2(d+1) = P_2(d) + r^*_{d+3L}\, r_{d+4L} - r^*_{d+2L}\, r_{d+3L} \tag{2.36}$$

The normalization term $R(d)$ is the received signal energy computed over the same sample windows as $P(d)$:

$$R(d) = \sum_{k=0}^{1} \sum_{m=0}^{L-1} |r(d + 2Lk + m + L)|^2 \tag{2.37}$$

Since this iterative form of (2.37) will play an important role, it can be specifically written as:

$$R(d+1) = R(d) + |r_{d+L}|^2 - |r_d|^2 + |r_{d+3L}|^2 - |r_{d+2L}|^2 \tag{2.38}$$

The timing metric takes the same normalized form as in the Schmidl–Cox method:

$$M(d) = \frac{|P(d)|^2}{(R(d))^2} \tag{2.39}$$

Similarly to the previous section, it is important to understand the relationship between the calculated metric and the preamble structure. As shown in Fig. 2.7, the maximum value of $M(d)$ corresponds to the very last sample of the preamble. Section 4 will compare the advantages and disadvantages of both methods in detail; for now, it is sufficient to keep in mind the general shape of the metric and its relationship to the received signal.

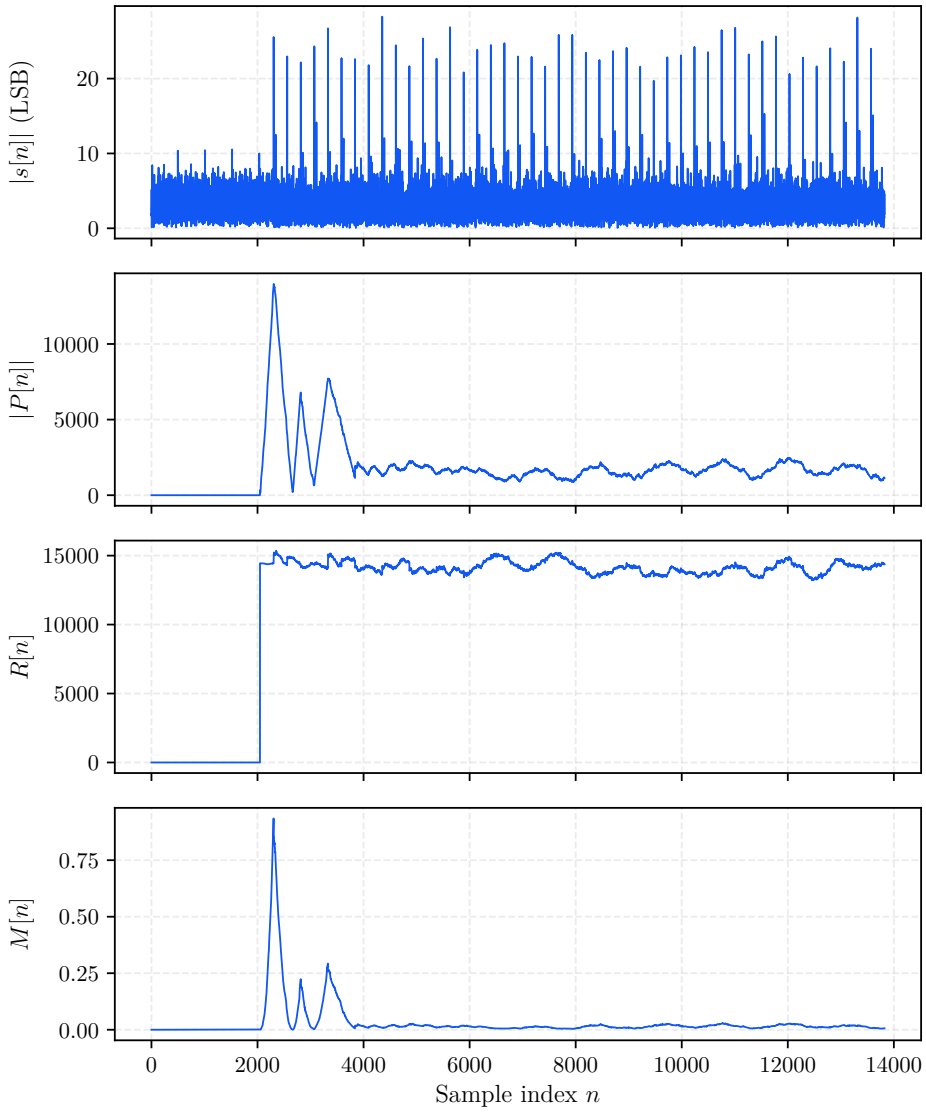

Figure 2.7: Minn metrics with the corresponding RX signal.

2.3.2.3 Radar Beamforming and Beam Steering

Beam steering is a signal processing technique that uses the capabilities of an antenna array to achieve directed signal beams. In close relation, beamforming is achieved by utilizing constructive and destructive interference patterns that can be used to form directive beams. For a normal single antenna, directivity can be achieved with a setup that includes a parabolic reflective dish and a motor to

physically direct the antenna to the desired angle. In applications where the user needs to direct the signal many times a second, mechanically redirecting the beam becomes impractical and sluggish. This is where beamforming and its applications of beam steering become very useful.

A phased array is a computer-controlled array of antennas where both the phase and amplitude of each array element is tuned to obtain a steerable main beam. The steerable beam is a designated main beam that is beamformed rapidly to be directed within the positional range of the designed system. The main beam is produced through constructive and destructive interference of the output from each element of the phased array. A series of phase shifter devices alter the phase of each individual antenna element. Control over the phase and gain of each element in tandem enables a dynamically steered beam controlled by the user, allowing the beam to be directed without the need for mechanical movement [22]. The fundamental principle behind beam steering is that applying a progressive phase shift across the array elements causes the beam to constructively interfere at a desired angle θ. The required phase shift applied to the n-th element is given by:

$$\phi_n = \frac{2\pi d}{\lambda} n \sin(\theta) \tag{2.40}$$

where d is the inter-element spacing, λ is the carrier wavelength, and $n = 0, 1, \ldots, N-1$ is the element index. The combined response of the array across all N elements is described by the array factor [23]:

$$AF(\theta) = \sum_{n=0}^{N-1} w_n \cdot e^{j \frac{2\pi d}{\lambda} n \sin(\theta)} \tag{2.41}$$

where w_n is the complex weight applied to each element, controlling both its phase and its gain. By adjusting these weights, the main beam can be steered electronically to any angle within the system's designed range.

In an analog beamforming architecture, phase shifts ϕ_n are applied directly in the RF domain using hardware phase shifters before the signals are combined. This means a single combined signal is passed to one receiver chain, which keeps hardware complexity low, but limits the system to forming only one beam at a time. Analog beamforming is the architecture relevant to the system, where the phase weights are set digitally but applied at the analog frontend. This is in contrast to digital beamforming, where the amplitude phase settings are applied directly to the digital signals before the DAC on transmit or after the ADC on receive.

A side effect of beamforming is the presence of side beams or "side lobes," which are usually nonessential beams at undesired directions and have considerably less field strength when compared to the main lobe. Phased array beamformers are popular for radar applications due to the electronic control of beam direction and are now used in 5G technology to direct communication signals to end-user devices. Leveraging the beamforming capabilities of this OFDM radar system enables it to have a much larger sensing range and the ability to test radar sensing and communications across different beam direction positions.

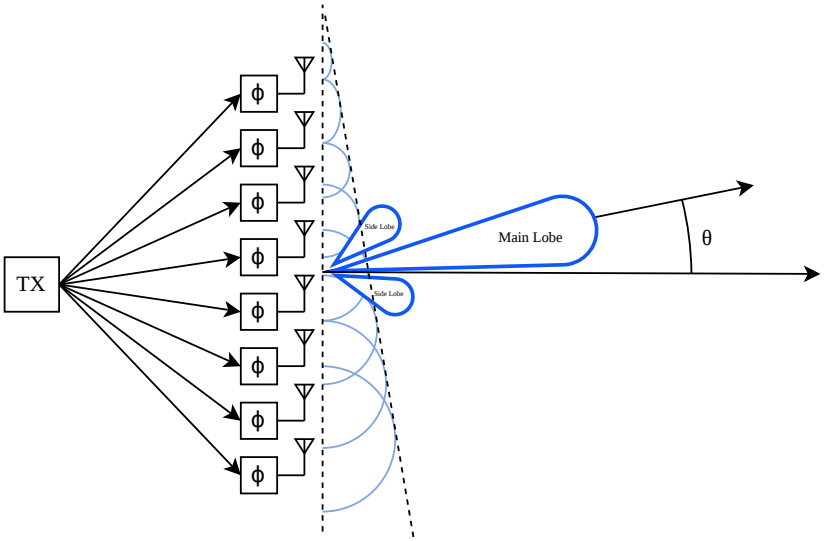

Figure 2.8: Phased array beamforming.

Chapter 3

Hardware System Design and Software Implementation

3.1 System Model

3.1.1 System Architecture Framework

The overall design is partitioned into four primary functional blocks. Each block is responsible for a distinct feature of the signal processing and transmission pipeline. At a high level, most of the changes and features were implemented in the receiver block where the synchronization and part of the data orchestration are located. The transmit block has a newly designed TX BRAM Reader to implement data replication and symbol handover. The beamformer controller remains largely unchanged except for hard wiring of the beam index return instruction. The data converters in the RFDC block remain largely unchanged from the previous iteration of the project. A complete system diagram can be found in *A.1*

3.1.1.1 Transmitter Block

The TX transmitter block generates the OFDM waveform and prepares the data for transmission through the RF frontends. QAM symbols are generated in the PYNQ notebook on the PS and transferred to the transmit block through a DMA connection. The TX BRAM Reader module enforces the described data frame structure, sequencing the preamble, LoS, and data symbols in the correct order. The frequency domain symbols are passed through the IFFT to produce the time domain OFDM structure, after which a cyclic prefix is prepended to each symbol to provide robustness against ISI. The resulting time domain symbols are forwarded to the RFDC for digital-to-analog conversion. In parallel, the LoS reference sym-

bols in the frequency domain are routed directly to the TX-RX interface RAM, bypassing the IFFT and transmission path.

3.1.1.2 Receiver Block

The RX block contains the majority of the novel hardware design elements developed in this work, with the exception of the TX BRAM Reader. The newly introduced functional IP blocks are the TX-RX interface RAM, the divider controller, the symbol counter, the CFO estimation and compensation block, and the Minn algorithm-based timing synchronization block. In operation, the received OFDM waveform is first digitized by the RFDC and passed to the timing synchronization block. The signal proceeds through the remainder of the receive chain only once the timing metric has been computed and its peak has been detected, indicating the start of a frame. In the same IP module, the CFO estimation and compensation are performed.

The cyclic prefix of each symbol is removed, and the data is transformed through the FFT IP block to recover the frequency domain representation of each received symbol. The resulting frequency domain symbol data is forwarded to the symbol counter and then to the divider block, where element-wise division with the corresponding reference symbols from the TX-RX interface RAM is performed. The symbol counter output ensures the correct alignment between the received and reference data streams at the divider input. Following this division, a separate IFFT is applied across the subcarrier dimension of each OFDM symbol to convert the channel transfer function estimate into the delay domain, producing the corresponding range profile. This IFFT is distinct from the modulation IFFT on the TX side and operates solely on the post-division frequency domain data. The range profiles are accumulated and transposed in preparation for a subsequent FFT applied across the slow-time dimension to resolve Doppler frequencies, yielding the range-Doppler map. This Doppler FFT is also distinct from the demodulation FFT earlier in the receive chain. The processed output is written back to the PS via DMA for post-processing, visualization, and analysis in Python.

3.1.1.3 Beamforming Controller

Beamforming control is an essential component of the RFSoC-to-RF frontend interaction, enabling the phased array antenna system to steer its beam toward a set of predetermined directions. These directions and the respective phase and gain characteristics for each phased array antenna are predetermined by Sivers in a proprietary beam book. The beam book organizes each beam direction with azimuth and elevation angle and its corresponding beam index.

The Sivers frontend receives instructions for rapid change of beam configuration or direction. The beam indices sequence is generated in the PS in the format required

by the Sivers product documentation and stored in BRAM. The generation and storage of the beam index sequence is done before the radar transmission and detection process. Once the transmission and detection process is initiated, the beam index sequence is decoded for each individual per symbol index and sent simultaneously with the reception of the OFDM symbol.

3.1.1.4 RFDC

The ADC and DAC configurations of the RFDC block have remained unchanged from the previous iteration of the project [2].

3.1.2 Novel OFDM Frame Structure

The data sent into the programmable logic, to the RF data converters, through the frontend modules, transmitted over the air, and received at the RX antenna, is organized in a unique way. The novel structure enables a timing synchronization algorithm feature and takes advantage of the system's beamforming capabilities by coordinating repeated LoS symbols with an antenna boresight beam position and subsequent data symbols over consecutive beam positions. The beam positions are arranged directionally with 21 evenly spaced angles in the azimuth direction and 3 evenly spaced angles in the elevation direction. This combination of azimuth and elevation directions results in 63 even separated angle permutations the narrow beam can occupy.

The preamble structure as described in Sections 2.3.2.2 is designated as the beginning of each frame, a frame consisting of a flexible number of symbols. The preamble symbol structure allows the system to know the exact timing synchronization for the start of each symbol. Each symbol that constructs the larger frame structure has a cyclic prefix prepended after the OFDM modulation IFFT in the TX block and will have the cyclic prefix removed before the OFDM demodulation FFT in the RX block.

The novel OFDM frame structure introduced in this work is designed to simultaneously support OTA timing synchronization, CFO estimation/compensation, beam formed spatial scanning, and radar processing within a coherent transmission scheme. The structure of the OFDM frame is organized as a hierarchy of data units, each serving a distinct role in the system. The different allocations and data hierarchy designations is as follows, along with Fig. 3.1 for visualization.

- Frames: A frame is defined as a concatenation of one preamble symbol followed by a configurable number of data "chunks."
- Chunks: Each chunk is composed of a single, repeating symbol iterated over a configurable duration to meet specific throughput or timing requirements.

- Symbols: In this system, each OFDM symbol consists of 256 words.
- Words: Utilizing a Super Sample Rate (SSR) 8 architecture, each word comprises 8 multiplexed samples. A single word has a total width of 256 bits, structured as a concatenation of eight 32-bit samples.
- Samples: Each sample is 32 bits. 16 bits for the real and 16 bits for the imaginary.

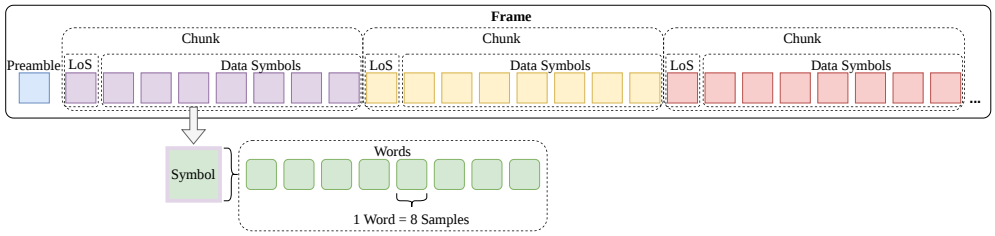

Figure 3.1: Data structure output of TX BRAM Reader.

The LoS symbol is a known reference OFDM symbol transmitted only at the antenna's boresight direction at the start of each data chunk. Since the receiver requires a clean reference to perform radar processing, the LoS symbol is captured and stored, and subsequent data symbols are divided against it to produce the radar matrix. The stored LoS symbols are congruent to the subsequent data symbols that make up the rest of the transmitted chunk. The LoS symbol and data symbols are transmitted and received in sequence by the frontends. Each LoS symbol is transmitted at the antenna boresight direction while the rest of the symbols are transmitted over the incrementing beam position sequence. A key part of this design that works in conjunction with the beamformer is the ability for the beamformer position and for the associated LoS symbols to periodically return to the antenna boresight position. On the receiver side, a symbol counter discards the incoming LoS symbols, while a divider control coordinates these symbols from the RX-TX interface RAM with processed data symbols to ensure accurate data division.

3.2 Hardware Setup

The complete hardware system comprises several hardware components working in tandem. Each component fulfills an important function to realize the functional OFDM ISAC radar system. The core of the system is the AMD ZYNQ UltraScale+RFSoC ZCU208 Evaluation Kit board. A crucial feature of this very large FPGA board is the eight integrated high speed 14-bit ADCs and eight integrated 14-bit DACs. The ZCU208 board runs on its own dedicated power supply and connects to the lab PC via Ethernet and Micro-USB. The lab PC serves as

the primary control interface, hosting the Jupyter notebooks used for system configuration, data transfer, and post-processing. The ZCU208 interfaces with the separate TX and RX EVK6005 frontends, with these units primarily responsible for the analog OFDM signal transmission and reception.

In the monostatic iteration of the project, a reference clock generator was used to supply both frontends with a synchronized clock signal. Removing this shared clock connection presented a significant challenge in creating a bistatic system setup and the EVK6005 frontend units were left to run on their own local oscillators. Each frontend unit utilizes its own power supply connection and Micro-USB connection to a USB splitter dongle on the ZCU208 board.

A "loopback" cable is installed as a bypass of the frontend transmission path for debugging purposes to test the system digital signal processing chain without any additional effects from the OTA transmission. The option to enable or disable OTA transmission using the loopback cable is controlled by the PYNQ Jupyter notebook. When stationary testing was performed in the designated lab space or mobile indoor testing, the hardware system was connected to the lab PC or one of the personal laptops for software processing and control using the PYNQ Jupyter lab.

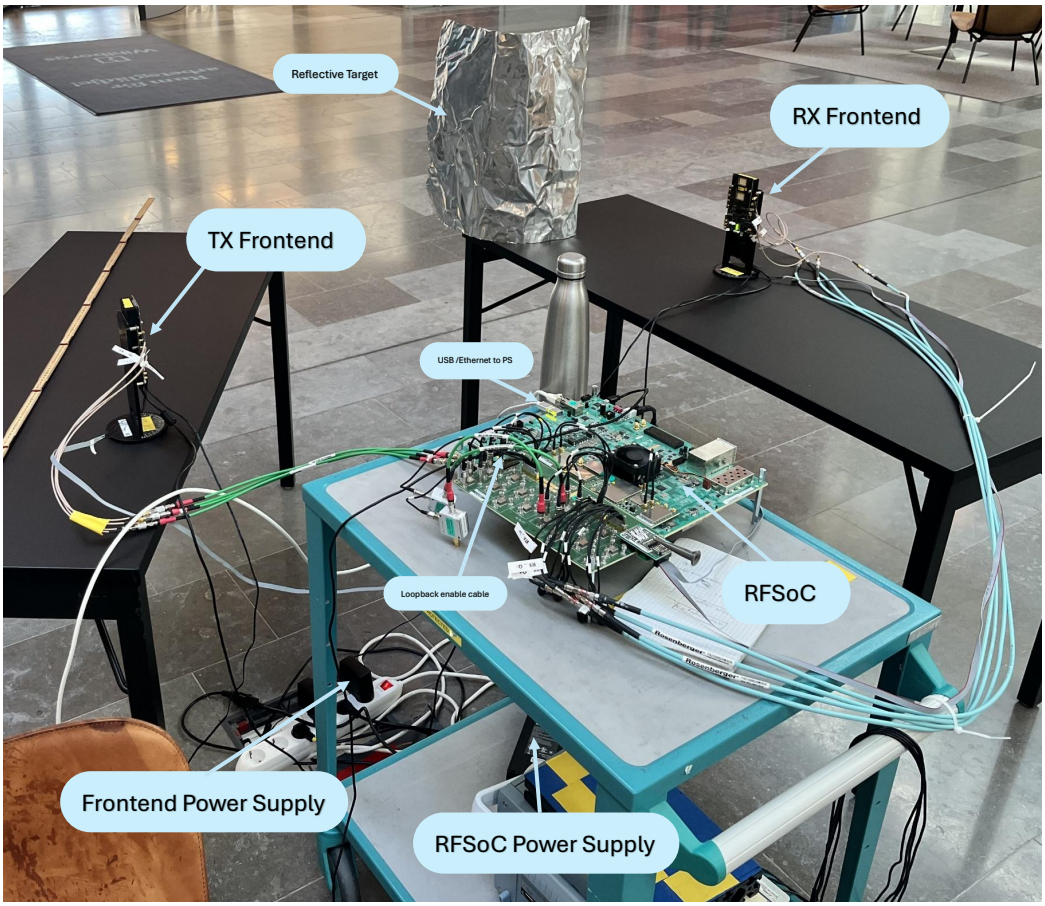

Figure 3.2: Hardware setup labeling

3.3 IP Blocks

3.3.1 Metric Calculation

The timing metric hardware matches the SSR feature of the system, processing eight samples in parallel on every clock cycle. The implementation computes $P(d)$ and $R(d)$ using a sliding window accumulator, as described in the iterative update equations from section 2.3.2.2.

3.3.1.1 Delay lines

The sliding window requires access to samples delayed by L, $2L$, and $3L$ positions. Since each clock cycle advances by the number of SSR samples, these delays correspond to L/SSR, $2L$/SSR, and $3L$/SSR clock cycles, respectively. Three sequential FIFOs provide the delayed sample vectors $r(n-L)$, $r(n-2L)$, and $r(n-3L)$.

3.3.1.2 Per-lane products.

The two correlation products use conjugate multiplication. Writing $r(n+k) = i_{0,k} + jq_{0,k}$ for the current sample and $r(n-L+k) = i_{1,k} + jq_{1,k}$ for the sample delayed by L, lane k contributes to P_1 via:

$$c_{1,k} = r^*(n-L+k)\cdot r(n+k) = \underbrace{(i_{1,k}\, i_{0,k} + q_{1,k}\, q_{0,k})}_{\text{Re}} + j\,\underbrace{(i_{1,k}\, q_{0,k} - q_{1,k}\, i_{0,k})}_{\text{Im}} \quad (3.1)$$

The same structure is used for $c_{2,k}$, which pairs the sample delayed by $2L$ with the sample delayed by $3L$:

$$c_{2,k} = r^*(n-3L+k)\cdot r(n-2L+k) \quad (3.2)$$

The two energy products require only magnitude-squared calculations, which are real-valued:

$$e_{1,k} = |r(n+k)|^2 = i_{0,k}^2 + q_{0,k}^2 \quad (3.3)$$
$$e_{2,k} = |r(n-2L+k)|^2 = i_{2,k}^2 + q_{2,k}^2 \quad (3.4)$$

Each clock cycle, therefore, produces four buses of eight products ($c_{1,k}$, $c_{2,k}$, $e_{1,k}$,

$e_{2,k}$ for $k = 0 \dots 7$). To prevent bit growth from propagating into the accumulator, each bus is reduced to a single value by a pipelined adder tree before being passed to the sliding window stage.

3.3.1.3 Sliding window accumulator

The iterative updates of (2.35) to (2.38) are implemented directly. A second set of FIFOs stores the SSR-wide partial sums from L/SSR cycles ago. On each valid cycle, the accumulator adds the incoming partial sum and subtracts the value read from the FIFO:

$$P_1(d + \mathrm{SSR}) = P_1(d) + c_{1,\mathrm{new}} - c_{1,\mathrm{old}} \tag{3.5}$$

and equivalently for P_2, R_1, and R_2. The four accumulators are then combined:

$$P(d) = P_1(d) + P_2(d) \tag{3.6}$$
$$R(d) = R_1(d) + R_2(d) \tag{3.7}$$

3.3.1.4 Metric computation

Once $P(d)$ and $R(d)$ are available, the timing metric $M(d) = |P(d)|^2/R(d)^2$ is computed in three pipelined stages:

1. Squaring: $|P(d)|^2 = P_{\mathrm{re}}^2 + P_{\mathrm{im}}^2$ and $R(d)^2$ are computed in registered multipliers.
2. Truncation: In both squared values, the upper 64 bits are retained while the lower 32 bits are discarded, preserving the dynamic range required by the divider while conforming to its input word width.
3. Division: An AMD Divider IP core computes the fixed-point quotient $M(d)$. Since $M(d) \in [0, 1]$, the quotient integer part occupies a single bit. The output passed downstream is formed by concatenating this least significant bit with the upper bits of the fractional result.

3.3.2 Peak Detection

The frame detector receives the timing metric $M(d)$ and the correlation metric $P(d)$ from the metric calculation stage and is responsible for identifying the timing instant of each received frame, which corresponds to the sharp peak in metric

$M(d)$. It outputs a sample index $\hat{d}$ and the corresponding $P(\hat{d})$ for use by the CFO estimator.

The detector is implemented as a three-state finite state machine (FSM) with states `IDLE`, `TRACK_MAX`, and `HOLD_OFF`.

The detector uses a two-stage process: it first detects when the metric exceeds a fixed threshold, and then tracks the maximum value within a bounded search window. This identifies the peak of $M(d)$ even in the presence of noise near the true timing point.

3.3.2.1 State Machine

1. `IDLE`:The FSM waits in `IDLE` until the incoming metric exceeds a fixed threshold M_{th}. On the first sample for which $M(d) > M_{\text{th}}$, the FSM records the current metric value and sample index as the initial maximum candidate and transitions to `TRACK_MAX`. The corresponding sample of metric $P(d)$ is also stored, since it is needed by the CFO estimator .

2. `TRACK_MAX`:The FSM scans a window of length equal to one frame size. On each cycle, the incoming metric is compared against the current maximum; if $M(d) > M_{\max}$ the maximum candidate, its index, and the associated $P(d)$ are updated. Once the window is exhausted the peak is confirmed, and the FSM asserts a flag.

3. `HOLD_OFF`: After confirming the peak, the FSM suppresses further detections for one frame period. This prevents the same frame boundary from being detected multiple times. After the hold off period, the FSM returns to `IDLE`.

3.3.3 CFO Estimation

The CFO estimation method used in this design is based on the approach introduced by Schmidl–Cox [4], which uses a deliberately structured preamble to convert an unknown frequency offset into a measurable phase angle. The key observation is that if the transmitted preamble contains two identical segments each of length L, then after passing through a channel with a CFO of ε, any two corresponding samples separated by exactly L positions will be related by a constant phase rotation. This follows from the fact that a CFO causes a linear phase ramp of $2\pi\varepsilon/N$ radians per sample, so over L samples the accumulated phase is:

$$\Delta\phi = \frac{2\pi\varepsilon L}{N} \tag{3.8}$$

In this design $N = 2048$ and $L = 512$, giving:

$$\Delta\phi = \frac{2\pi\varepsilon \times 512}{2048} = \frac{\pi\varepsilon}{2} \tag{3.9}$$

Taking the conjugate product of each pair of corresponding samples yields a complex number pointing in the direction $e^{j\Delta\phi}$. Since this phase is identical for every pair, summing L such products coherently preserves the phase while reinforcing the magnitude, so the phase of the resulting metric equals $\Delta\phi$ directly:

$$P(d) = \sum_{k=0}^{L-1} r^*(d+k)\, r(d+k+L) \tag{3.10}$$

where $r(n)$ is the received sample sequence and d is the candidate timing position. The phase of this metric therefore equals $\Delta\phi$ directly:

$$\angle P(d) = \Delta\phi = \frac{\pi\varepsilon}{2} \tag{3.11}$$

$P(d)$ serves a dual purpose: its magnitude is used for timing synchronization, while its phase directly encodes the CFO. Once the timing peak is detected, the complex value of $P(d)$ [24] is passed to a downstream Coordinate Rotation Digital Computer (CORDIC), which computes the four-quadrant arctangent of its imaginary and real parts. Since $\angle P(d) = \pi\varepsilon/2$, inverting this relationship yields the fractional CFO estimate:

$$\hat{\varepsilon} = \frac{2}{\pi}\angle P(d) = \frac{2}{\pi}\arctan\left(\frac{\mathrm{Im}\{P(d)\}}{\mathrm{Re}\{P(d)\}}\right) \tag{3.12}$$

The acquisition range of this estimator is bounded by $|\hat{\varepsilon}| < 1$ subcarrier spacing, which is an inherent consequence of the $(-\pi, +\pi]$ output range of the arctangent operation.

3.3.3.1 Hardware Implementation

The hardware implementation is built around a CORDIC IP core configured in translation (arctangent) mode. The IP iteratively rotates the input vector toward the positive real axis using a sequence of power-of-two angle steps, converging to the four-quadrant arctangent. The correlation metric $P(d)$ is com-

puted by the timing synchronization unit and passed as a pair of signed integers ($\mathrm{Re}\{P(d)\}$, $\mathrm{Im}\{P(d)\}$) to the estimator. The estimator then performs three operations in sequence: input bit-width reduction, arctangent computation, and a fixed-point shift that converts the raw angle into the per-sample phase increment forwarded to the compensator. The CORDIC IP with a 24-bit output width represents phase angles in a signed Q3.21 fixed-point format. The integer value 2^{21} corresponds to 1.0 in the Q3 integer field, and IP maps π to:

$$\pi \equiv 2^{21}, \qquad 2\pi \equiv 2^{22} \tag{3.13}$$

The arctangent output is confined to $(-\pi, +\pi]$, which occupies the central half of the 24-bit signed code range. A general angle $\theta \in (-\pi, +\pi]$ is therefore encoded as:

$$\theta_{\mathrm{raw}} = \left\lfloor \theta \cdot \frac{2^{21}}{\pi} \right\rceil \tag{3.14}$$

where $\lfloor \cdot \rceil$ denotes rounding to the nearest integer. Substituting equation 3.14 into equation 3.12:

$$\hat{\varepsilon} = \frac{2}{\pi} \cdot \theta_{\mathrm{raw}} \cdot \frac{\pi}{2^{21}} = \frac{\theta_{\mathrm{raw}}}{2^{20}} \tag{3.15}$$

This value is represented as an integer quantity in the fixed-point implementation. $\hat{\varepsilon}$ is then converted into the signed per-sample phase step applied by the compensator to correct the CFO.

3.3.3.2 Numerical Example

To connect the practical scenario of the earlier section directly to hardware register values, the normalized CFO of $\varepsilon = 0.063$ is traced through every stage of the estimation pipeline.

Phase angle presented to the CORDIC.

$$\angle P(d) = \frac{\pi\varepsilon}{2} = \frac{\pi \times 0.063}{2} \approx 0.09896 \text{ rad} \tag{3.16}$$

CORDIC output is:

$$\theta_{\text{raw}} = \frac{\pi \times 0.063}{2} \times \frac{2^{21}}{\pi} = 0.063 \times 2^{20} \approx 66{,}060 \text{ LSBs} \tag{3.17}$$

At this point, it should be noted that the compensator accumulates phase in a 24-bit register in which a full 2π rotation corresponds to 2^{24} LSBs, so a per-sample increment of one subcarrier spacing ($\Delta\phi = 2\pi/N = 2\pi/2048$) is represented as $2^{24}/2^{11} = 2^{13}$ LSBs. Since the CORDIC output encodes $\hat{\varepsilon}$ as $\theta_{\text{raw}} = \hat{\varepsilon} \cdot 2^{20}$ (3.15), scaling to the compensator's units requires multiplying by 2^{13} and dividing by 2^{20}, which is a net right-shift of $20 - 13 = 7$ bits.

$$\Delta\phi_{\text{hw}} = -(\theta_{\text{raw}} \gg 7) = -\left\lfloor \frac{66{,}152}{128} \right\rfloor = -516 \tag{3.18}$$

The relationship between this shift and the system parameters is:

$$\frac{\theta_{\text{raw}}}{128} = \frac{\theta_{\text{raw}}}{2^7} = \hat{\varepsilon} \cdot \frac{2^{20}}{2^7} = \hat{\varepsilon} \cdot 2^{13} \tag{3.19}$$

The negation produces the conjugate rotation required to cancel the positive frequency offset. The magnitude of 516 LSBs can be verified against the theoretical value from (3.18):

$$|\Delta\phi_{\text{hw}}|_{\text{theory}} = \varepsilon \cdot 2^{13} = 0.063 \times 8{,}192 = 516.1 \tag{3.20}$$

The results of (3.20) and (3.18) show that, in general, a small residual CFO will remain uncompensated due to fixed-point quantization. This residual manifests as a slow phase drift over time, with a correspondingly larger effect on longer frames. Therefore, the best achievable hardware estimate for the example above is 516. The potential deviation between this optimal hardware-representable value and the value achieved in simulation, along with its practical impact, will be examined in Section 5.

3.3.4 CFO Compensation

Once the CFO has been estimated, the received signal must be corrected. This is achieved by multiplying each received sample $r(n)$ by a complex exponential whose phase is the conjugate of the accumulated CFO phase, producing a corrected sample:

$$\tilde{r}(n) = r(n) \cdot e^{-j\,2\pi\hat{\varepsilon}n/N} \tag{3.21}$$

where n is the absolute sample index from the start of the data chain. The per-sample phase increment of this rotation is:

$$\delta\phi = -\frac{2\pi\hat{\varepsilon}}{N} \tag{3.22}$$

The negation ensures the rotation cancels the accumulated phase.

3.3.4.1 Hardware Implementation

The hardware compensator matches the Super Sample Rate (SSR) of the system, processing SSR = 8 samples in parallel on every clock cycle, meaning the phase accumulator must advance accordingly. The compensator receives the per-sample phase increment $\Delta\phi_{\text{hw}}$ from the estimator, encoded in a 24-bit unsigned format in which a full 2π rotation corresponds to 2^{24} LSBs. The per-sample increment of one subcarrier spacing is therefore:

$$\delta\phi_{1\text{sc}} = \frac{2^{24}}{N} = \frac{2^{24}}{2^{11}} = 2^{13}\ \text{LSBs} \tag{3.23}$$

and $\Delta\phi_{\text{hw}} = \hat{\varepsilon} \cdot 2^{13}$ as derived in the estimation section. In the clock cycle, when a valid phase increment arrives, the accumulator is initialized to the phase already accumulated during the preamble.

$$\phi_{\text{acc},0} = (N + CP) \cdot \Delta\phi_{\text{hw}} \tag{3.24}$$

Since estimation is done at the very end of the preamble, this ensures phase continuity, so that the first data sample is corrected with the phase it has accumulated. On every subsequent valid data cycle, the accumulator advances by the SSR step:

$$\phi_{\text{step}} = \text{SSR} \cdot \Delta\phi_{\text{hw}} = 8 \cdot \Delta\phi_{\text{hw}} = \Delta\phi_{\text{hw}} \ll 3 \tag{3.25}$$

Within a single SSR clock cycle, the eight parallel lanes correspond to consecutive samples $n, n+1, \ldots, n+7$. Lane k therefore requires an additional phase offset of:

$$\phi_{\text{offset},k} = k \cdot \Delta\phi_{\text{hw}}, \qquad k = 0, 1, \ldots, 7 \tag{3.26}$$

The total phase applied to lane k on a given cycle is:

$$\phi_k = \phi_{\text{acc}} + \phi_{\text{offset},k} \tag{3.27}$$

Each lane is corrected by a second AMD CORDIC IP core, this time configured in rotation mode. The CORDIC expects its phase input in a signed 24-bit format in which $\pi \equiv 2^{21}$, i.e. $2\pi \equiv 2^{22}$. Since the accumulator uses $2\pi \equiv 2^{24}$, the lane phase must be right-shifted by 2 before being presented to the CORDIC:

$$\phi_{\text{cordic},k} = -\left(\phi_k \gg 2\right) \tag{3.28}$$

The negation is the hardware implementation of the conjugate rotation presented in equation 3.21. The CORDIC then correctly rotates the complex input sample.

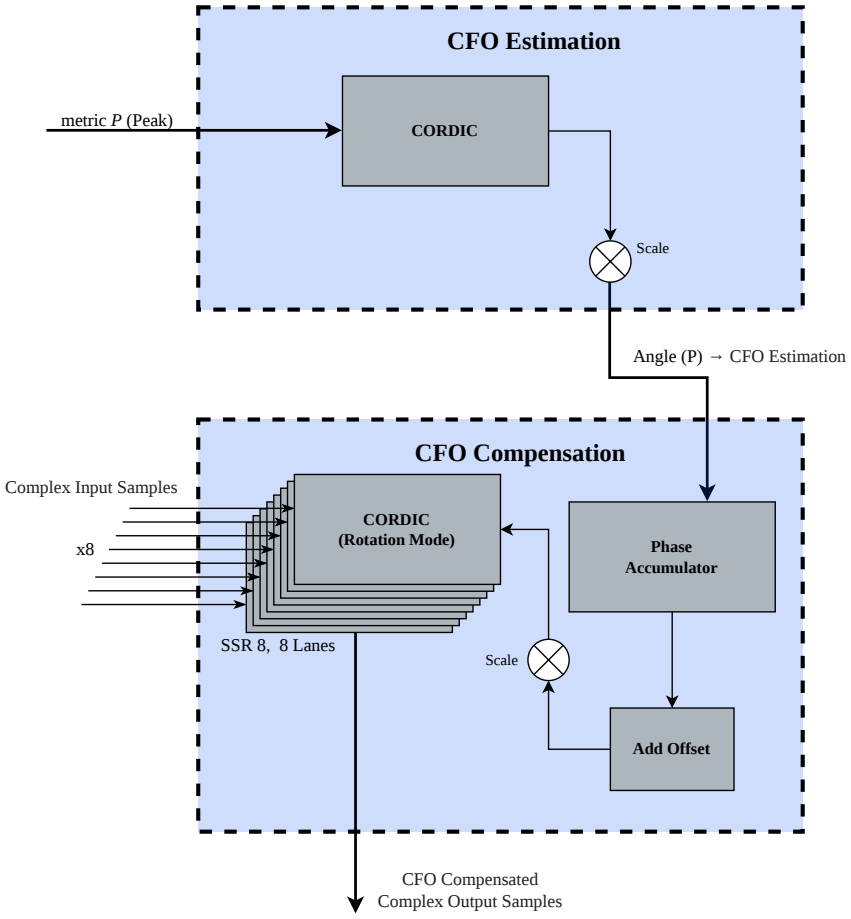

Figure 3.3: CFO estimation and compensation block diagram.

3.3.5 Top Level Integration

The processing blocks described in the preceding sections are assembled into a single top level module that implements both the synchronization as well as CFO correction. The module accepts a raw SSR-wide complex sample stream and produces a phase corrected output stream aligned to the detected frame boundary. Fig. 3.4 shows a block diagram of the module.

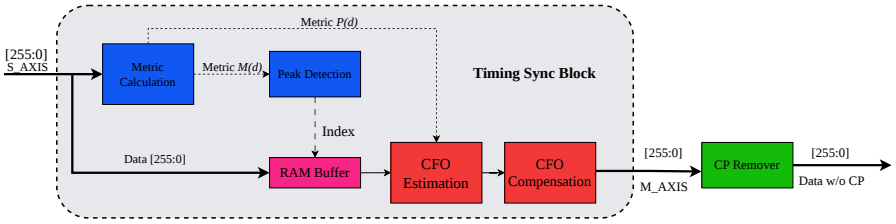

Figure 3.4: Top level Minn-based timing synchronization and CFO estimation/ compensation.

The overall signal flow proceeds as follows. The incoming sample stream is broadcast to two parallel paths simultaneously: the metric calculator, which processes it in real time to produce the timing metric and corresponding metric $P(d)$, and a set of eight per lane circular RAM buffers, which store every arriving sample so that the relevant portion of the frame can be replayed once the frame boundary has been identified.

3.3.5.1 $P(d)$ Alignment Pipeline

The timing metric $M(d)$ passed to the frame detector is the result of a division, implemented using a Divider IP core, which introduces additional pipeline latency. Since $P(d)$ does not pass through the divider and therefore arrives at the frame detector input earlier than the $M(d)$ sample it corresponds to. For compensation, $P(d)$ samples are passed through a shift register before being presented to the frame detector. This delay matches the latency of the divider and a few extra pipeline stages in the $M(d)$ path, so that the $P(d)$ value arriving at the frame detector is always aligned with its corresponding $M(d)$ sample. The frame detector then registers the correct $P(\hat{d})$ when the peak is confirmed.

3.3.5.2 Sample Buffer and Address Calculation

Eight single-port RAM modules, one per SSR lane, form a circular buffer that continuously records the incoming stream. When the frame detector asserts its output, the top-level FSM latches both the current write address and the detected peak offset, and computes the RAM read address from which replay should begin:

$$read\ address = write\ address - pipeline\ latency + peak\ offset \tag{3.29}$$

Where the total pipeline latency from the input to the write address register, accounting for the timing metric calculation, frame detector, and alignment pipeline delays.

The output is controlled by a six-state FSM that coordinates all the features mentioned above.

3.3.6 TX BRAM Reader

The main role of the TX BRAM reader is to read data from the Direct Memory Access (DMA) in the PS, write the data downstream to BRAM, and send the data downstream to the RX data path and TX data path. A key design requirement for the TX BRAM Reader is its ability to replicate symbols based on a predetermined configurable frame structure. The IP from the previous iteration of the project [2] originally had a TX BRAM Reader IP that simply wrote DMA data from the PS into a main TX block BRAM and read out the data to be passed down the TX data chain or RX side divider for direct data division. The necessity of the new TX BRAM Reader is for data playback features that will repeat symbols to fit the desired data frame structure. The first full symbol of each chunk that comprises the frame is captured by the `los_symbol_capture` module. Each LoS symbol is saved in the smaller TX RX interface RAM module to be divided against transmitted and received data to create range-Doppler maps.

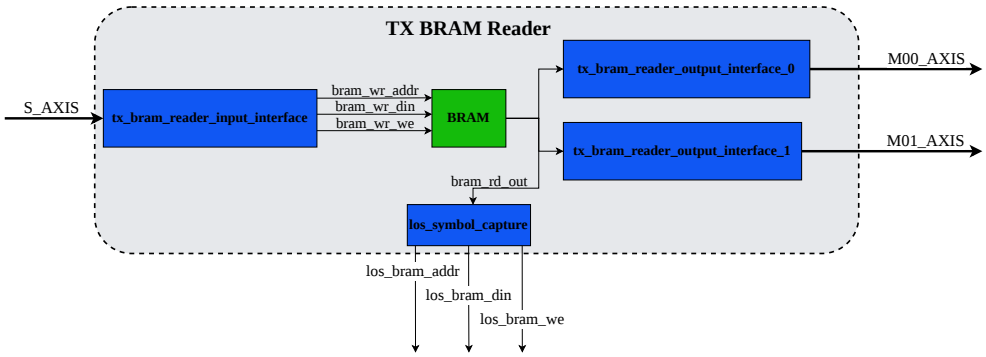

Figure 3.5: TX BRAM Reader block diagram.

The IP is comprised of 4 main submodules that handle data write into the BRAM, read from the BRAM, and symbol capture storing for later radar processing. The input interface handles the AXI handshake and writes incoming words sequentially into the BRAM write port. Two identical output interfaces handle read from the BRAM readout ports and carry those to the rest of the TX data chain, and a now unconnected divider reference. The LoS symbol capture submodule passively snoops the read data path for the first symbol of each transmitted chunk, storing it into a separate external BRAM module for use as a radar division feedback reference during RX processing.

The top-level module is configured through an AXI-Lite register interface, through which the Processing System (PS) sets the runtime parameters *nword* and *nloop*. These will govern the number of words per transfer and the number of playback iterations of the TX BRAM Reader, respectively. A set of IP configuration parameters further defines the operating characteristics of the block, including the word width in bytes, BRAM depth, BRAM read latency, words per OFDM symbol, symbol repeat count, LoS symbol storage depth, and the LoS BRAM address width. These parameters allow the block to be adapted to a range of OFDM frame configurations without requiring hardware restructuring.

Table 3.1: TX BRAM Reader Parameter Table

Parameter	**Value**
Word Bytes	32
Parameter Width	32
BRAM Depth	32768
BRAM Latency	128
Words Per Symbol	256
Symbol Repeats (How many symbols are repeated to complete a chunk)	9
LoS Symbol Depth (Number of complete symbols stored within the RAM module	4
LoS BRAM Addr Width	10

3.3.7 Symbol Counter

The symbol counter tracks the position of each new symbol within a chunk. Based on the counter value, two flags are derived:

- `is_los` is asserted during the first symbol of each chunk, indicating that the current symbol is the LoS reference.
- `is_data` is asserted for all remaining symbols, identifying them as data symbols for further processing.

A constant defining number of symbols per chunk allows the module to be reconfigured for different frame structures without modifying the design.

3.3.8 Divider Control

The divider control module is responsible for synchronizing the incoming data symbols with the corresponding LoS reference stored in memory, so that the matching pair is presented to the downstream frequency domain divider.

3.3.8.1 Output Interface

The aligned pair is presented to the divider on two independent channels:

- **dividend**: the data symbol received through the channel and processed on the receiver side.
- **divisor**: the unprocessed reference symbol from the transmitter side.

3.3.8.2 LoS Address Tracking

Since channel estimation and compensation are outside the scope of this project, therefore when the symbol counter asserts `is_los`, the module only has to record the base memory address for the current LoS symbol:

$$base\ address = current\ symbol \cdot number\ of\ words\ per\ symbol \tag{3.30}$$

A write address cycles through the available memory slots, advancing by one each time a complete LoS symbol is received. Holding multiple LoS symbols in memory simultaneously is necessary because the write and read operations are separated by the combined latency of the transmit pipeline, receiver processing, and the signal path between transmitter and receiver.

3.3.8.3 Data Path and RAM Read Alignment

When the symbol counter asserts `is_data`, the module drives the RAM read address word by word for the duration of the symbol. At the end of each data symbol, the read address is reset to the base address in preparation for the next symbol.

To account for the fixed latency introduced by the sequential logic following the external memory, the incoming data is delayed by two clock cycles before being presented as the dividend. The memory output therefore, arrives aligned with the delayed data word, ensuring a coherent pairing between dividend and divisor.

3.3.9 TX-RX Interface RAM

This RAM module IP is used to interface between the incoming LoS symbol data from the transmit block and pass over the reference to the receive block. The interface RAM is configured to store four complete symbols at a time to handle latency of the transmit and receive data paths, and the complete OTA transmission time. The Divider Control IP reads repeated symbols as orchestrated by the symbol counter for each chunk length, ceasing to read symbol data from the interface RAM if *is_los* is asserted. The read address is passed from the output of the divider control module. The *write enable*, *write address*, and *data in* are all directly connected from the output of the TX BRAM Reader.

Chapter 4

Methodology

An important early decision that influenced both the development process and the final results was the choice of tools and methods. Since RTL development is time-consuming and demands a high degree of precision, beginning with a high level reference model was a natural choice. This approach significantly accelerated the evaluation of candidate algorithms and provided greater confidence when the RTL design stage began. The reference model also proved invaluable during debugging, as will be discussed later in this section.

For high level modeling, the two most prominent options considered were MATLAB together with Simulink, which provides a range of relevant built-in libraries [25], and Python [26] [27], which is an open-source tool offering straightforward setup and seamless integration with the user's preferred development environment and version control system. A key advantage of Python in this context is its compatibility with the PYNQ framework used by the hardware platform, meaning that code developed and validated during the early stages of the project could be transferred directly to the system with minimal modification.

The choice of hardware toolchain was largely constrained by the existing design, which had already been developed targeting an AMD FPGA, making Vivado the only viable option. The development continued in Vivado 2023.2 to maintain consistency with the existing project. Although the original design had been written in Verilog, VHDL was chosen for this work based on personal preference. This introduced no meaningful drawback, as Vivado supports mixed-language designs natively.

4.1 Algorithmic Exploration

To evaluate candidate timing synchronization algorithms, a Python-based reference model was developed covering the full signal chain: OFDM frame assembly at the transmitter, propagation through a configurable channel model, and reception including synchronization, CFO compensation, and data extraction.

Based on the literature survey, two algorithms were selected for detailed evaluation. The Schmidl–Cox algorithm served as the primary benchmark. The range of available resources, theoretical analyses, and practical implementations provided a solid foundation for evaluation and comparison. The Minn algorithm was selected as the main contender, offering a similar computational structure while addressing the most significant limitation of the Schmidl–Cox method. Other algorithms [28] [29] [30] were considered as candidates for either standalone coarse timing synchronization or as a refinement stage, but were discarded early in the evaluation phase due to excessive computational overhead or insufficient synchronization performance under the target channel conditions. The following sections therefore focus on the two selected algorithms and their performance across a range of channel environments and configurations.

4.1.1 Preamble Generation

- **Schmidl–Cox Preamble:** To achieve the repetition in the time domain Schmidl–Cox preamble is generated in the frequency domain by placing a pseudo-noise sequence symbols on the even subcarriers and setting the odd subcarriers to zero [4]. The IFFT then produces a time domain symbol whose first and second halves are identical, satisfying the structure required by the Schmidl–Cox correlation. Later, a cyclic prefix is added.
- **Minn Preamble:** Following the same principle, every fourth subcarrier is assigned a binary phase value of ± 1, while the remaining subcarriers are set to zero. The signs are drawn from a balanced sequence, containing equal numbers of $+1$ and -1 values to ensure that the two sub-correlations in $P(d)$ contribute equally, maximizing the sharpness of the timing metric peak. [16] The IFFT of the sparse frequency-domain sequence returns a time-domain signal, to which a cyclic prefix can be added.

4.1.2 Channel Model

To assess synchronization performance under realistic conditions, the reference model supports several channel impairments:

- **Additive White Gaussian Noise (AWGN):** Complex Gaussian noise

with configurable variance σ^2 is added to the received signal.

- **Symbol Timing Offset (STO):** An arbitrary number of noise samples is prepended to the transmitted signal, shifting the frame start.
- **Carrier Frequency Offset (CFO):** A frequency offset ε is applied as a phase rotation $e^{j2\pi\varepsilon n/K}$, modeling oscillator mismatch between transmitter and receiver.
- **Sampling Frequency Offset (SFO):** A relative clock difference δ between transmitter and receiver is modeled by resampling the transmitted signal onto a stretched time axis, followed by truncation to the original length.
- **Multipath:** A target list specifies per-path delay, Doppler shift, and complex amplitude. Each path is applied as a fractional frequency domain time shift, with optional Rayleigh fading for distributed clutter targets.

4.1.3 Frame Detection

Once the correct OFDM frame structure was assembled, the timing metrics $P(d)$, $R(d)$, and $M(d)$ were computed for both the Schmidl–Cox and Minn methods, as shown in Fig. 2.5 and 2.7. The Schmidl–Cox algorithm served as the primary benchmark, chosen for its well-documented performance and suitability for hardware implementation [31] [32]. As noted in the previous chapter, the two methods produce metrics with distinctly different shapes, and therefore require different post-processing steps to reliably identify the start of the data symbols. A frame detection algorithm was implemented for each method to evaluate both the accuracy and the complexity of extracting the frame boundary from the respective metric.

4.1.3.1 Schmidl–Cox Frame Detection

The plateau in the Schmidl–Cox timing metric means the peak of $M(d)$ cannot be used directly as a timing estimate. Instead, the following post-processing steps are applied [33]:

- **Smoothing:** The metric $M(d)$ is passed through a moving-average filter, producing a smoothed metric $\tilde{M}(d)$. This smooths the plateau and reduces noise-induced fluctuations.
- **Differentiation:** The discrete derivative $\Delta(d) = \tilde{M}(d+1) - \tilde{M}(d)$ is computed. The edge of the plateau corresponds to a zero-crossing of $\Delta(d)$ from positive to negative.
- **Thresholding:** A zero-crossing is accepted only if the corresponding metric value exceeds a threshold γ, suppressing false detections due to noise.

- **Blanking:** After each accepted detection, a blanking window is applied to prevent the same plateau from triggering multiple detections.

These additional steps add algorithmic complexity and introduce sensitivity to the choice of threshold and filter length, but the algorithm still produced consistently reliable frame detections.

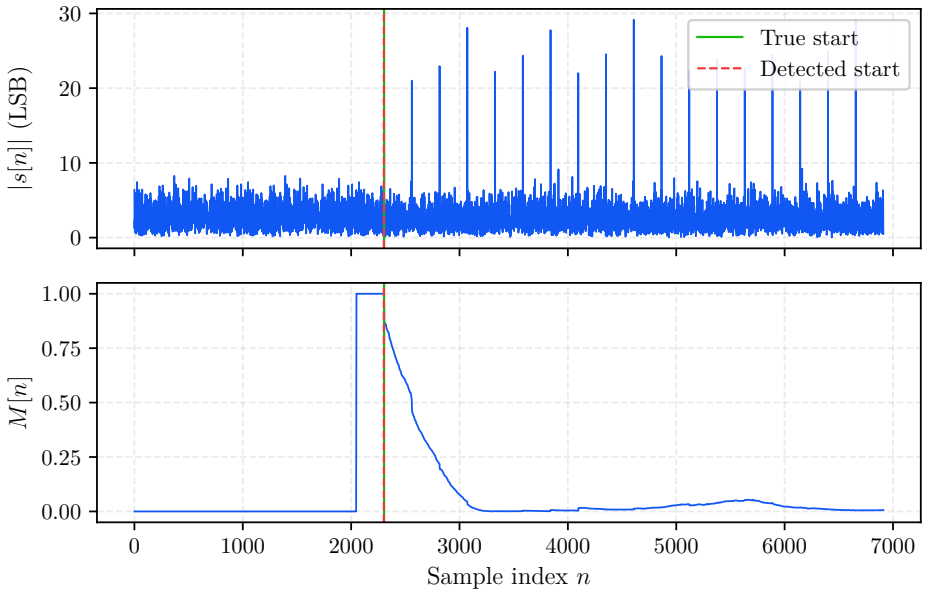

Figure 4.1: Timing estimation based on Schmidl–Cox metric.

4.1.3.2 Minn Frame Detection

Unlike the Schmidl–Cox method, no additional post-processing is required since the metric exhibits a sharp, well-defined peak at the correct timing point. Therefore, the corresponding last sample of the preamble is:

$$\hat{d} = \arg\max_{d} \; M(d) \tag{4.1}$$

Following (4.1), the beginning of the frame can be estimated.

4.1.4 Schmidl–Cox Algorithm Performance

The algorithm was evaluated under a representative set of channel conditions. The noise variance was set at $\sigma^2 = 0.1$ (SNR $\approx 20\,\text{dB}$), the SFO at $\delta = 1\,\text{ppm}$, consistent with the CFO analysis of Section 2.3.1, and the multipath channel was

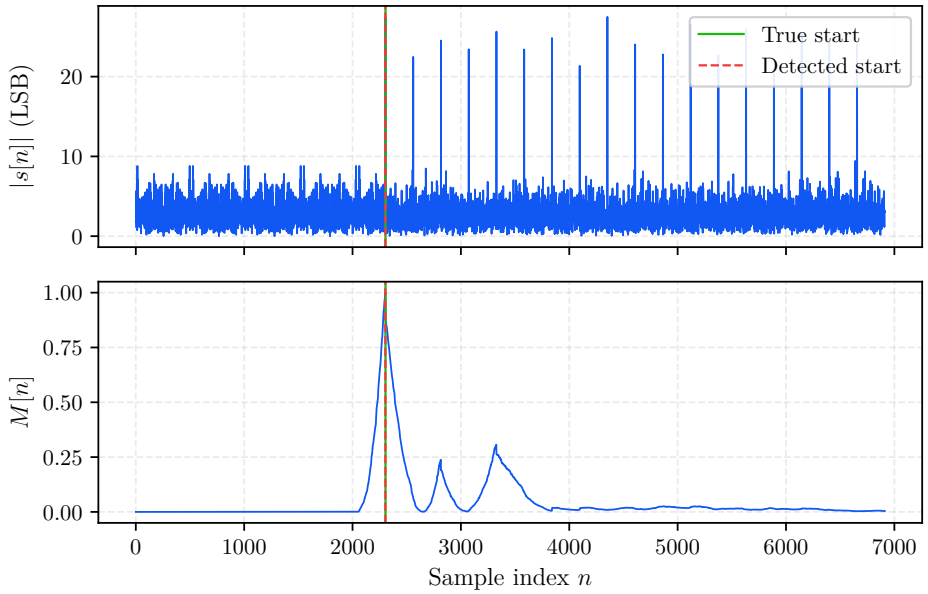

Figure 4.2: Timing estimation based on Minn metric.

configured with two reflectors at path lengths of 6 m and 15 m, corresponding to propagation delays of 20 ns and 50 ns, respectively. No CFO compensation was used for this section.

Under these conditions, the algorithm located the frame boundary with a timing error of two samples. Even when taking the SSR = 8 parallel processing structure into account, the error remained well inside the cyclic prefix interval, meaning ISI is therefore avoided and the error has no effect on the recovered data.

4.1.5 Minn Algorithm Performance

The Minn algorithm was evaluated under the same channel conditions as the Schmidl–Cox algorithm to allow a direct comparison.

The Minn algorithm reduced the timing error to one sample under identical conditions. In addition to the marginal improvement in timing accuracy, the Minn metric does not require additional post-processing to resolve the peak location. Based on these results, the Minn algorithm was selected for hardware implementation.

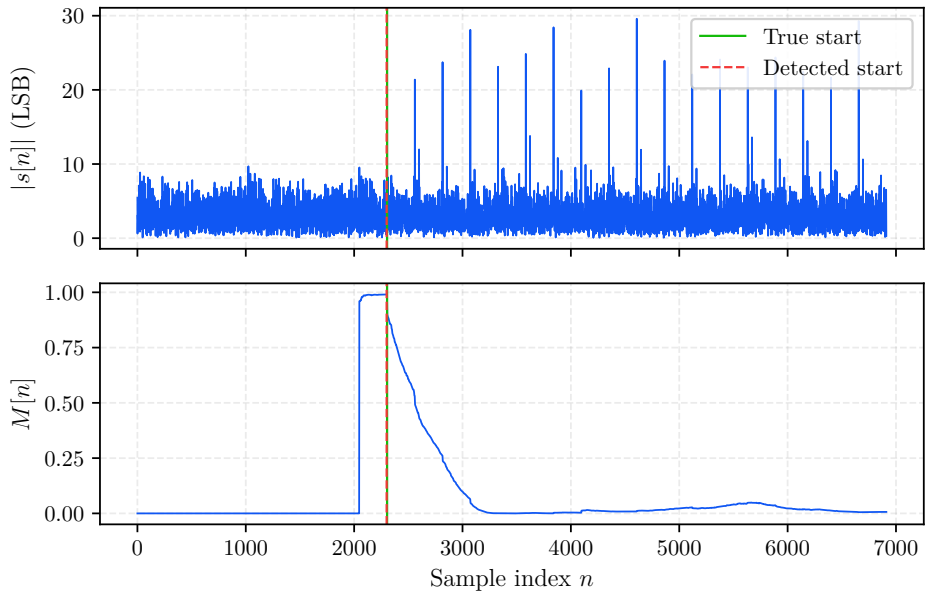

Figure 4.3: Schmidl–Cox timing metric $M(d)$ and detected frame boundary under the simulated channel conditions.

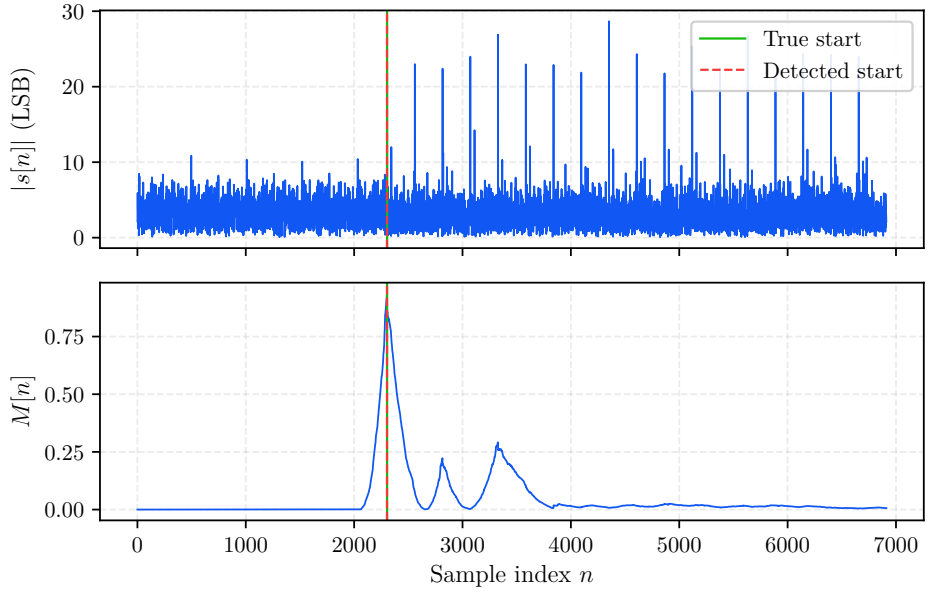

Figure 4.4: Minn timing metric $M(d)$ and detected frame boundary under the simulated channel conditions.

4.2 Alternative Bistatic Synchronization Scheme

A bistatic radar setup with variable beamforming position capabilities introduces an interesting opportunity to test and evaluate different synchronization schemes with different physical setups. With 2D beamforming capabilities with the chosen RF frontends it is possible to select a specific direction for the synchronization symbol beam angle. For a baseline case, the boresight direction was chosen with 0 ° azimuth and 0 ° elevation. With the TX and RX frontends directly facing each other, the synchronization signals and the LoS symbols are ensured to be sent in the most direct path. The other beam directions are then used for sensing objects around the beam direction 2D scan. This configuration was used as a baseline scenario as it was predicted to have the best synchronization and radar performance. This scheme is presented as a bistatic radar system in Fig. 2.3

A novel setup to test synchronization is to use the side lobes for timing synchronization instead of only relying on the main beam and varying the beam directions. The side lobes of the directed pencil beams lie at a predictable acute angular offset. Although the side lobes operate at a much lower amplitude in comparison to the main lobe, the viability of using the side lobes for timing synchronization was investigated. This setup is potentially advantageous, as all beam directions could be used for sensing instead of reserving a direction and many transmitted symbols for timing synchronization. If timing synchronization could be achieved using the side lobes, this would enable different possibilities for varying and more flexible bistatic radar setups.

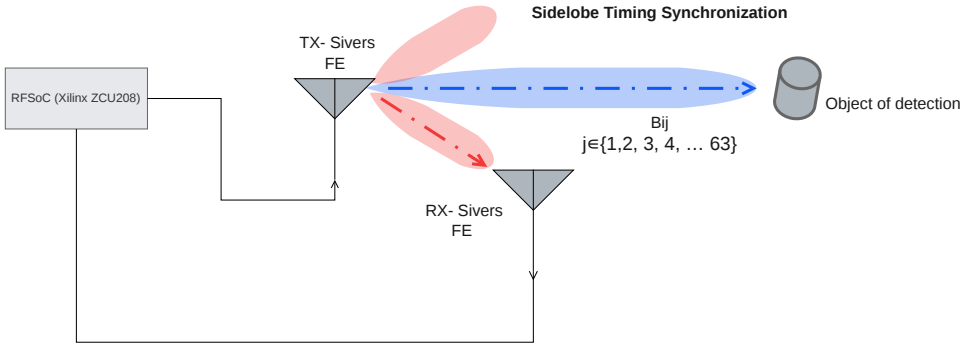

Figure 4.5: Sidelobe synchronization geometry.

4.3 Hardware Design and Implementation

With the high level reference model established, the focus shifted to the hardware design. This phase involved defining the state machine architecture, optimizing resource utilization, and evaluating timing accuracy. As the monostatic design

already consumed the majority of the available on chip memory, this represented an obvious bottleneck that constrained the RTL design from the outset.

The remaining components described in Section 3.3 followed a similar verification flow. After each block had been validated in isolation, all components were brought together in a unified simulation environment. The final integration test was designed to verify the end-to-end functionality and timing of the system by comparing the two inputs to the Divider Control block. Using Python, two stimulus files were generated: one in the frequency domain, which was fed directly into the TX BRAM Reader, and a corresponding time-domain file that replicated the TX processing chain by matching the repeating symbol structure produced by the TX BRAM Reader, followed by the IFFT and CP addition. A fully connected single processing chain could not be established in simulation due to difficulties configuring the CP Adder in the simulation environment. The time domain stimulus was therefore constructed to approximate the CP adder's output. The time domain path was then passed through the CP Remover, Integer Padding block, FFT, Symbol Counter, and finally the Divider Control. Under ideal conditions, with no OTA channel effects applied, the two inputs to the Divider Control are expected to be closely matched. A small residual difference remained due to the processing applied on the receive side, which was anticipated. Nevertheless, the test confirmed that the timing and functional behavior of the integrated system are correct and that the design is ready for deployment on the hardware platform.

Once all components had been verified both individually and in combination, they were packaged as custom IP blocks in Vivado and integrated into the main design.

4.4 Hardware System Platforms

4.4.1 Hardware Software Interaction and Interface, PL and PS

The system was built on the AMD Zynq UltraScale+ RFSoC ZCU208 hardware platform, which combines standard FPGA programmable logic (PL) with an integrated multicore ARM Cortex-A53 processor subsystem (PS). The PL is a reconfigurable hardware part of the chip, as in any modern FPGA, it consists of configurable logic blocks (CLBs), programmable interconnects, and other hardware components such as DSP slices, block RAM (BRAM) and I/O pins. This part is used for custom hardware acceleration and handles all of the hardware functions. These include the timing synchronization, metric calculation, CFO estimation and compensation, cyclic prefix addition and removal, FFT and IFFT operations, beamforming control, and the TX BRAM Reader data sequencing logic.

The processor system in contrast, includes a general purpose processor that ex-

ecutes software, mostly handling high level control, such as user interfaces and system management. Data transmission between the two systems relies mostly on direct memory access (DMA) and memory mapped I/O (MMIO). For heavy data transmissions, that require high throughput, the system utilizes DMA, with beamforming index control handled via the MMIO interface, due to low latency requirements of the beamforming controller in the PL. Specific user set parameters such as *nword* and *nloop* settings of the TX BRAM Reader are set using the MMIO interface as well.

The system also utilizes PYNQ [34], an open source framework that exposes hardware control through Python and Jupyter Notebooks. PYNQ provides a set of Python libraries that enable direct access to DMA and MMIO registers, allowing seamless configuration and control of the hardware design from a high level environment. This provides a flexible prototyping workflow with fast iteration and straightforward system evaluation.

4.4.1.1 Memory Optimization

As indicated by the utilization figures in Table 4.1, the monostatic design operated close to the limits of the available on chip resources, particularly with respect to the memory. Accommodating the additional logic required by the bistatic extension therefore had to target memory optimization without compromising the core processing functionality.

Table 4.1: Hardware utilization comparison

	Monostatic Utilization [%]	Quasi-Bistatic Utilization [%]
LUT	61	64
LUTRAM	16	15
FF	49	50
BRAM	91	83
URAM	100	100
DSP	33	35
IO	3	3
BUFG	1	1
MMCM	13	13

One feature of the monostatic design is a dedicated RX time domain path, which branches off at the very beginning of the RX pipeline and bypasses all subsequent processing. This path exists as a diagnostic channel, providing direct visibility into the raw complex data samples for inspection purposes. Because the ADC continuously produces samples while the DMA is controlled via software and only begins accepting data when explicitly triggered, a buffering stage is required to

bridge the gap between sample arrival and DMA readiness. In the original design, this was realized as a deep FIFO chain, which consumed a significant portion of the available BRAM. Since the time domain path is not essential for the radar's core function, the FIFO depth was reduced to free on-chip memory for the bistatic components. This introduces a marginal risk of sample loss at the very start of a capture window due to DMA initialization latency but has no effect on the Doppler processing chain. The resulting utilization figures for the quasi-bistatic design are summarized in Table 4.1, where the BRAM utilization has been reduced from 91% to 83%.

4.4.2 RF Frontend Devices

The system uses EVK06005 RF frontend evaluation kits operating at 60 GHz. Each kit integrates a BFM06009 beamforming module capable of 2D beam steering with rapid beam configuration switching. While each frontend includes both a TX and RX chain, the current bistatic configuration dedicates one unit exclusively to transmission and the other to reception.

Beamforming is achieved through digitally controlled phase shifters and gain blocks within the antenna array, enabling precise steering of the beam toward specific angular sectors. By sequentially directing the beam across a set of predefined angles within a single measurement frame, the radar captures spatial diversity information that complements the range and Doppler signatures. Beam switching is controlled by the FPGA via dedicated GPIO lines, synchronized with the radar timing logic.

4.5 Data Collection and Analysis

For system performance evaluation, data was collected using both simulations and an Integrated Logic Analyzer (ILA). The ILA is an IP core used to monitor internal signals of the FPGA design in real time. Its main features include Boolean trigger equations and edge transition triggering. Data captured by the ILA can be exported in comma separated value (CSV) format and analyzed using tools such as Python. However, the ILA has notable limitations: the number of samples that can be captured is constrained by the on chip memory available, and the presence of hardware and channel effects can obscure certain signal characteristics, making some results harder to interpret. For these reasons, simulation results are presented in selected cases throughout the following section, as they allow cleaner isolation of individual design features.

4.6 Experimental Setup

The physical setup of the radar system has several iterations and different environments based on the needs of testing and the environmental needs of the tests being performed.

For physical system testing of the hardware, different physical setups and environmental conditions were used for both ease of testing and for the ability to test different features of the radar system independently from each other. All testing was either conducted in the lab bench space designated for this project team or the open lobby of the Ericsson office. The need for periodically moving the entire hardware system was to mitigate close-by interference objects such as metallic furniture, computer equipment, and a large glass partition. This type of environment introduces a degree of multipath reflections and background clutter into the received signal. Thus, radar performance testing was completed in an open environment where these multipath effects and background clutter could be mitigated. Measurements were taken when no human movement was present in the lobby space.

The complete hardware setup was mobile as all components could be stored on a wheeled cart. This made setup and transportation from the lab bench to the open lobby straightforward. For lobby testing, most of the hardware was centrally positioned on the cart platform, and the RF frontends were positioned at a predefined distance away, facing each other for the bistatic setup. The centrally positioned ZCU208 RFSoC evaluation board was connected via Ethernet and USB to a laptop running the PYNQ Jupyter Lab environment. The laptop or desktop PC served as the primary control and data processing interface, handling system configuration, data capture, and visual post-processing in Python.

Three distinct physical configurations were used across the testing campaign, each fulfilling a distinct purpose.

4.6.1 Lab Bench Experiment Setup

When development was ongoing, some developmental functional verification was done in the lab to test features. These included beamforming indexing or confirming that timing synchronization was working for different project configurations. The cart sat in the corner of the lab space while the RF frontends were positioned a meter apart. This length was limited by the physical confines of the lab space itself. An optional loop back cable was used instead of OTA transmission for testing that involved isolation of results from the highly reflective multi-path environment. Testing transmission tests only using the loop back configuration served as a baseline validation for the digital signal processing chain. The direct

coaxial cable would bypass the analog RF transmission path and frontend antenna arrays entirely. The loop back path was enabled and disabled through a PYNQ notebook software flag.

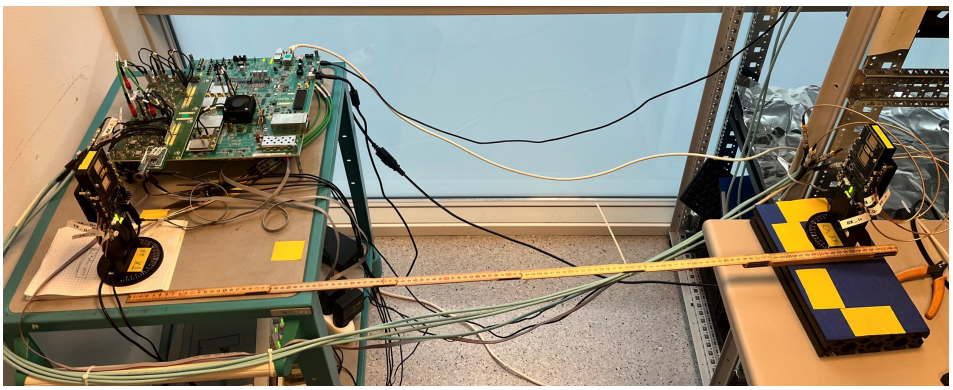

Figure 4.6: Limited lab bench physical setup.

4.6.2 Quasi-Bistatic Experiment Setup

The primary OTA testing configuration placed the TX and RX frontend units facing each other directly across the bench at a separation distance ranging from 1.4 to 2 meters. Both units were positioned at the same height and oriented in such a way that their antenna boresight directions were aligned at the direct path between the two units. This maximized the received signal strength on the LoS beam direction.

For target testing, a reflective metal sheet was used as the sensing target during performance evaluation. The sheet was placed at varying distances from both frontends. The results of this testing are described in the results chapter. This configuration served as the primary evaluation setup for both timing synchronization performance under OTA conditions and radar sensing capabilities, including the generation of the range-Doppler maps.

Figure 4.7: Quasi-bistatic experiment setup

4.6.3 Side Lobe Synchronization Configuration

A secondary OTA configuration was used to investigate whether timing synchronization could be achieved using the side lobes of the transmitted beam rather than solely relying on the main lobe, as motivated in Section 4.2. In this setup, both frontend units were initially placed approximately 30 cm apart on the bench, with each frontend facing the same general direction. Different angular orientations between the TX and RX frontends were tested to realize the threshold angle where timing synchronization would be successful. No sensing targets were present during these trials to isolate the synchronization behavior of the system. A target induced reflection would otherwise trigger a false timing synchronization event.

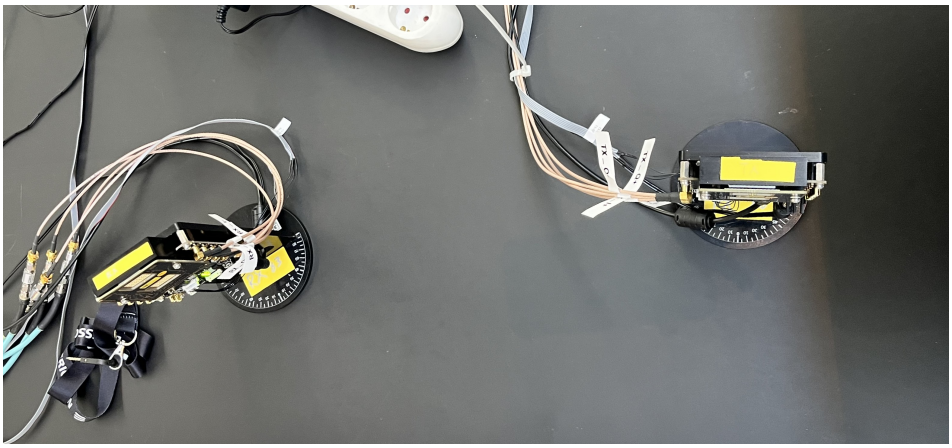

Figure 4.8: Side lobe synchronization setup

Chapter 5

Results and Analysis

Converting a monostatic radar setup into a quasi-bistatic configuration introduces significant challenges, most notably the need for synchronization between the decoupled TX and RX units, as well as compensation for the unwanted effects that arise as a direct consequence of the decoupling. Addressing these challenges was the primary objective of this work. Specifically, the focus was on achieving reliable timing synchronization and on estimating and compensating for carrier frequency offset, which was identified as the main target impairment. The performance of both features is evaluated in the following chapter along with analysis of the baseline radar performance and side lobe synchronization.

5.1 Timing Synchronization

To critically evaluate the over-the-air synchronization, a performance metric must be defined. The primary limiting factor in timing synchronization is the CP length. If the timing offset exceeds the CP length, ISI is introduced and recovered data becomes corrupted. Therefore, provided the synchronization error remains below 144 samples, equal to the CP length in the implemented system, the received data can be correctly recovered at the RX side.

For the synchronization performance evaluation, the high level implementation results were presented in the previous chapter. Here, the RTL implementation is assessed through both simulation and hardware measurements captured via the ILA during live over-the-air testing.

5.1.1 Simulation Results

For detailed simulation testing, the same Python model was used to generate a variety of input stimulus files, which were then fed into the RTL model. Different channel conditions were tested, including varying levels of CFO and SFO. The RTL model matched the reference well, showing accurate performance under ideal conditions and only minor deviations when additional impairments were introduced. The main performance difference inherent to the hardware model comes from the SSR architecture of the system. Since eight samples are processed in parallel, the synchronization precision is limited to eight samples. As a result, a one sample timing offset in the reference model corresponds to an eight sample offset in hardware.

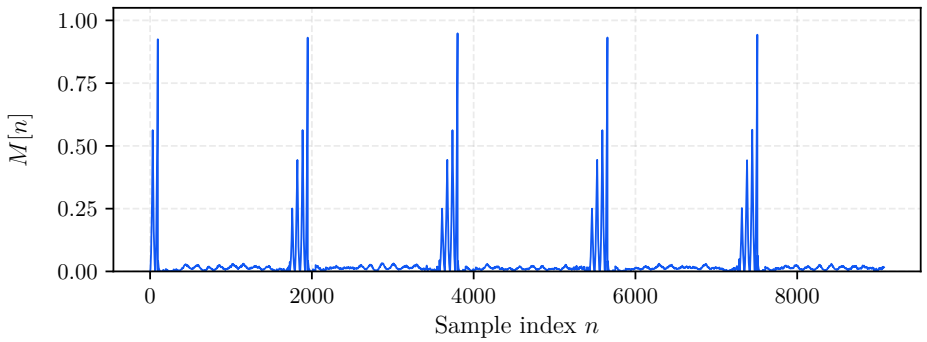

Figure 5.1: Timing metric calculated in RTL for five consecutive frames.

As shown in Fig. 5.1, the timing metric is correctly computed for each individual frame. Following metric calculation, the frame detector operates in conjunction with the synchronization logic to align the data stream with the peak detection output. To validate the end-to-end functionality, the input and output data samples are compared directly. Since the position of the first data sample in the generated input is known, any timing offset or data mismatch can be detected unambiguously. It is important to note that the preamble symbol is discarded at the receiver and does not appear in the output data stream. Therefore, the preamble must also be excluded from the input reference file prior to comparison, so that the sample indices are correctly aligned and the comparison yields meaningful results. At this stage, no CFO compensation is applied, so the data stream passes through unmodified. Consequently, when timing synchronization is correct, the output samples are expected to match the input exactly (excluding preamble). This was confirmed consistently across a range of simulated channel conditions, demonstrating robust and reliable timing synchronization performance of the RTL model.

5.1.2 Hardware Results

Timing performance can also be evaluated in hardware by using ILAs to compare data captured on the TX side, which serves as the reference, against the output of the timing synchronization module. Since the LoS beam direction is used to transmit the preamble, which is always followed by the first data symbol also sent on the direct beam, no timing discrepancy is expected between these first two symbols. As both are transmitted on the LoS beam, they have a minimized impact by reflections from surrounding targets. Two peak detector parameters proved to be the most critical for timing estimation: the threshold value, which determines when the frame detector transitions from `IDLE` to `TRACK_MAX`, and the window length, which defines how long the detector remains in the `TRACK_MAX` state. As shown in Fig. 5.2–5.6, different combinations of parameters yield noticeably different results.

Through both high level Python simulations and RTL simulations, a threshold of 0.3 was found to be optimal. The simulations were conducted under a variety of conditions, including noise, CFO, SFO, and multipath channel effects. A lower threshold caused the state transition to trigger too early, resulting in the timing metric peak being missed. A higher threshold did not yield improved results in either simulation or hardware testing, and was therefore dismissed, as it risks missing the peak entirely under real channel conditions.

All examples use a threshold of 0.3, while the window length is varied. In both unsuccessful timing estimations, shown in Fig. 5.4 and 5.5, a shorter window length equal to half a symbol period was used, which proved insufficient. Using a threshold of 0.3 combined with a window length of one full symbol period, the timing estimation was consistently accurate, as shown in Fig. 5.2 and 5.3. Results are presented for two cases: with the full beamforming scan pattern active, and with a fixed beam direction index. Each figure marks the frame start to indicate where data symbols truly begin. Early samples collected at the first instance of `TVALID` from the timing synchronization module reveal a slight mismatch between the true and received frame starts. However, these extra samples from an early timing detection are acceptable as they fall within the CP length of 144 samples.

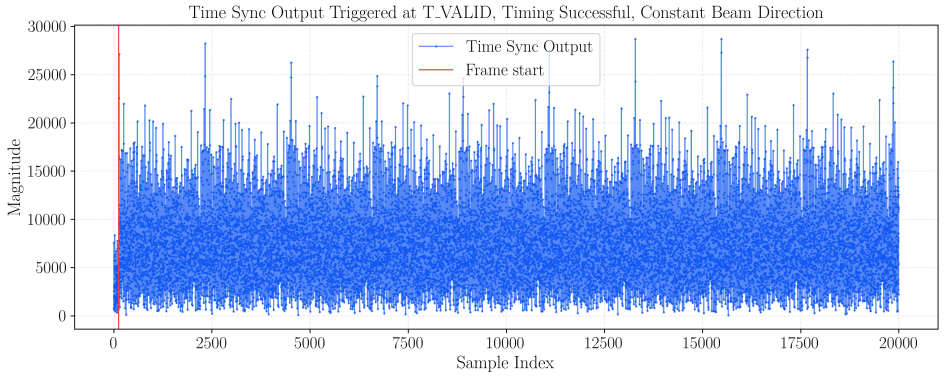

Figure 5.2: Time sync module output. Timing successful. Testing using a fixed beam direction for consistent amplitudes across each signal with a 0.3 threshold value.

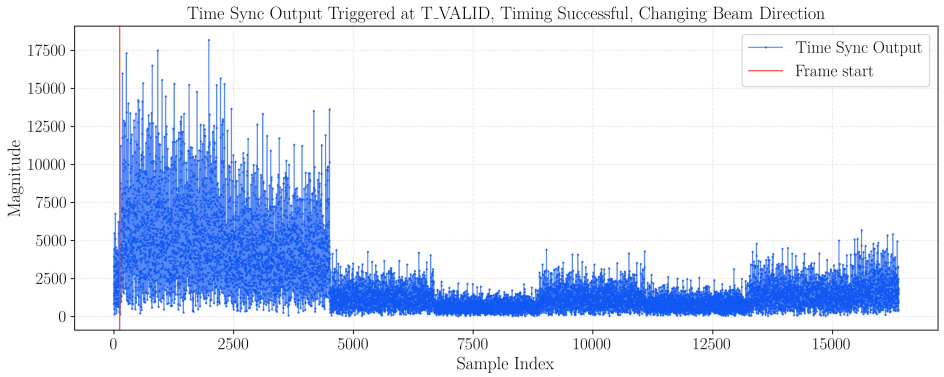

Figure 5.3: Time sync module output triggered at the first instance of TVALID. Timing Successful. Testing using changing beam directions per symbol with a 0.3 threshold value. Output starts correctly at the first intended symbol and not during the preamble symbol.

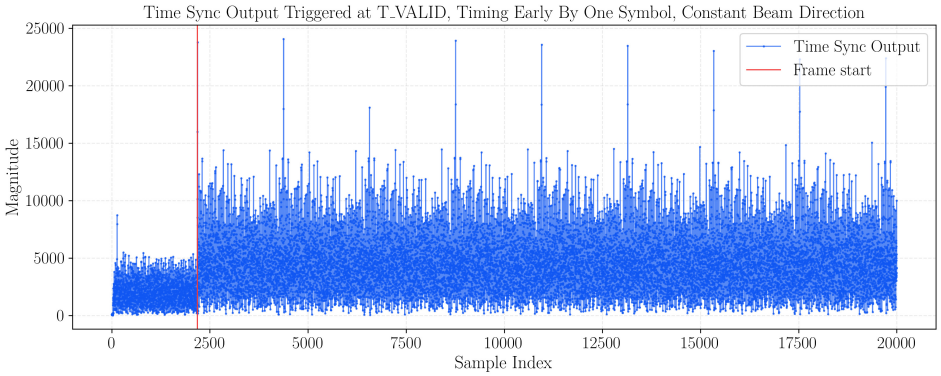

Figure 5.4: Time sync module output triggered at the first instance of TVALID. Timing unsuccessful as an amount of the preamble symbol is present. Testing using a fixed beam direction for consistent amplitudes across each signal with a 0.3 threshold value.

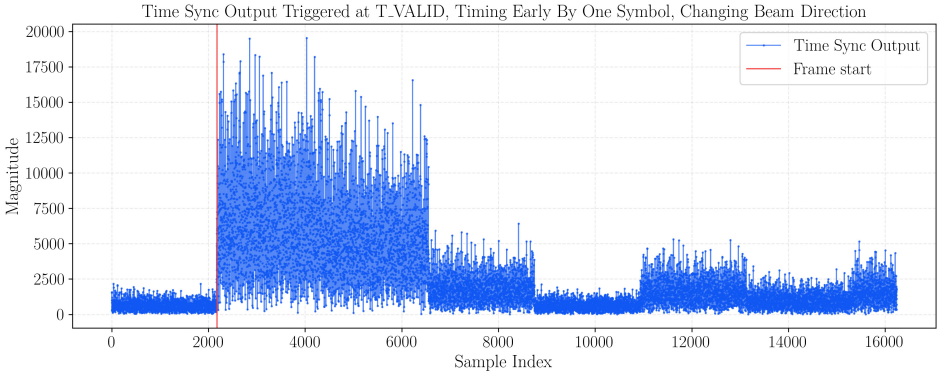

Figure 5.5: Time sync module output triggered at the first instance of TVALID. Timing unsuccessful as an amount of the preamble symbol is present. Testing using changing beam directions per symbol with a 0.3 threshold value.

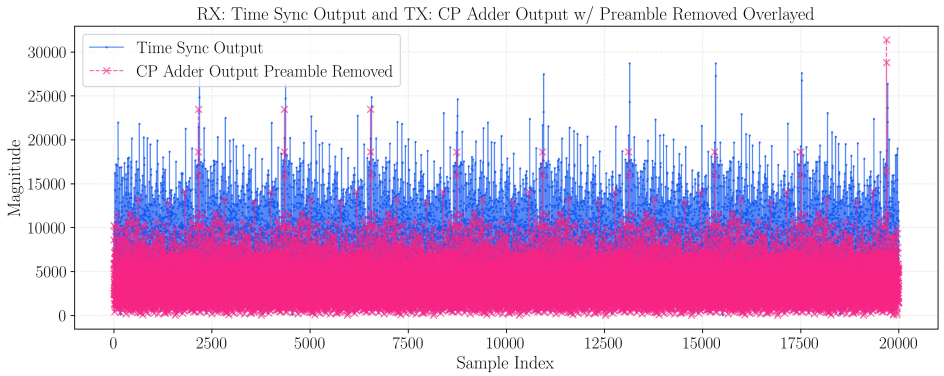

Figure 5.6: Outputs samples overlay of the output of time sync module in the RX block and CP adder module output in the TX block triggered at the first instance of TVALID. Preamble symbol length has been removed from the CP adder module output as to compare per symbol synchronization directly.

One of the main objectives of this work was to develop a sufficiently robust synchronization method that can be applied in various configurations, as described in Section 4.2. Thus far, all tests were conducted in the forward scatter configuration [35]. When setting up the configuration where side lobes are used for synchronization, all tests were performed without a target present, as reflections from a target could trigger a successful synchronization. Since the synchronization must operate reliably regardless of the environment, a setup without a target was used to isolate the synchronization behavior.

The tests began with the TX and RX units placed approximately 30 cm apart, both facing the same direction, essentially mimicking a monostatic configuration. This setup did not result in successful synchronization. The RX was then progressively tilted towards the TX, and consistent synchronization was achieved at an angle of approximately 140°. This confirms that the beam side lobes can be used for synchronization, while the main part of the beam can be used for sensing. A successful synchronization is shown in Fig. 5.8, where the frontends are positioned as shown in Fig. 4.8

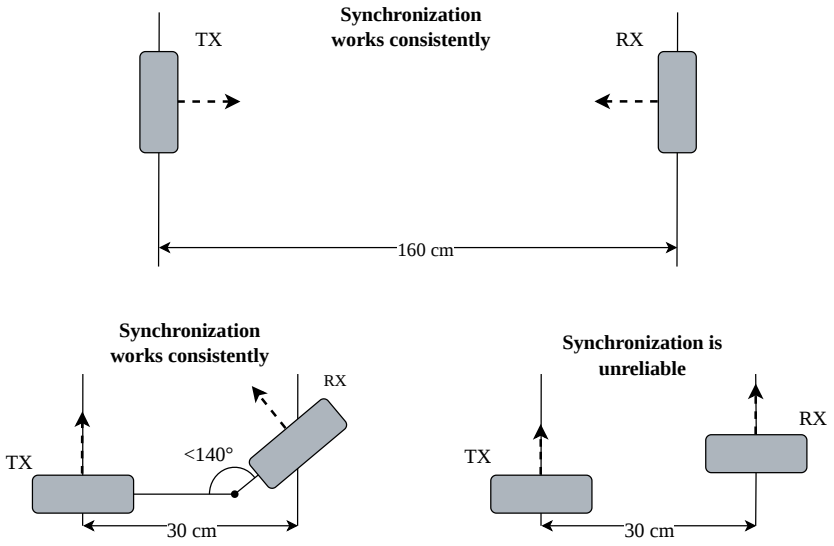

Figure 5.7: Side lobe synchronization testing setup.

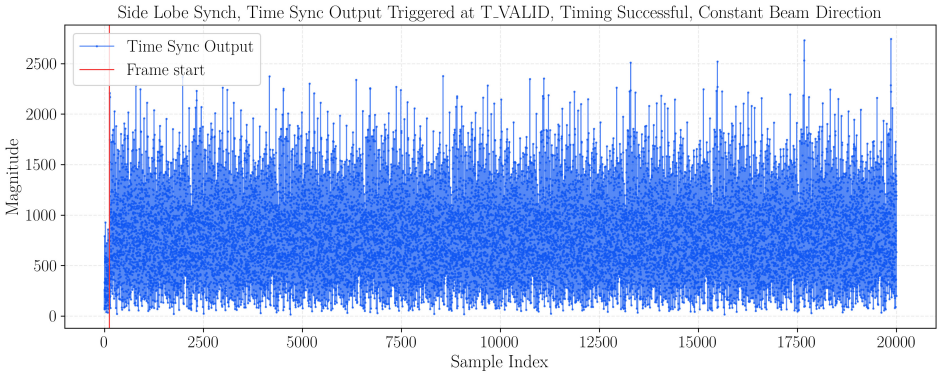

Figure 5.8: Side lobe synchronization testing. Time sync module output, timing successful. Testing using a fixed beam direction for consistent amplitudes across each signal with a 0.3 threshold value.

5.2 CFO Estimation and Compensation

For the CFO estimation and compensation results, simulation data is primarily presented. Over-the-air transmission introduces additional impairments that make isolated CFO evaluation difficult. Therefore, a controlled simulation environment

allows for more accurate and unambiguous validation. For this purpose, both a clean reference signal and a CFO corrupted version were generated and fed into the hardware design. The compensated output was then compared against the clean reference to confirm correct compensation. In all simulations, the CFO value applied in Python matches the value derived in Section 3.3.3.

5.2.1 Simulation Results

As discussed previously in the Section 3.3.3, the closest hardware representable approximation to a normalized CFO of 0.063 corresponds to a phase increment value of 516. This value was also produced by the hardware estimation module, confirming that the estimator output matches the expected result.

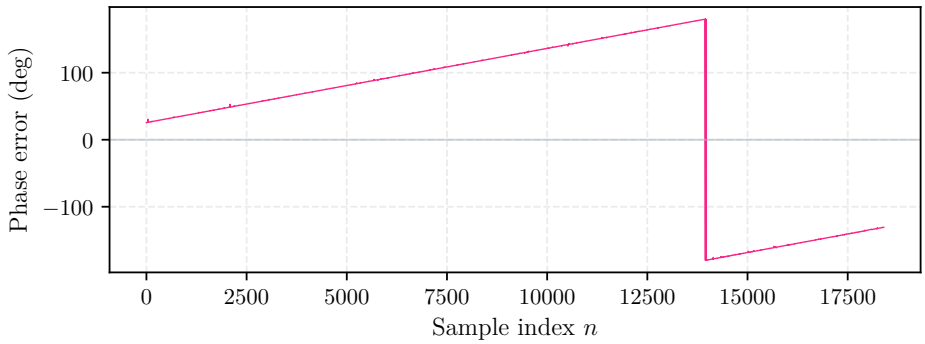

Figure 5.9: Error in amplitude and phase between clean and corrupted data.

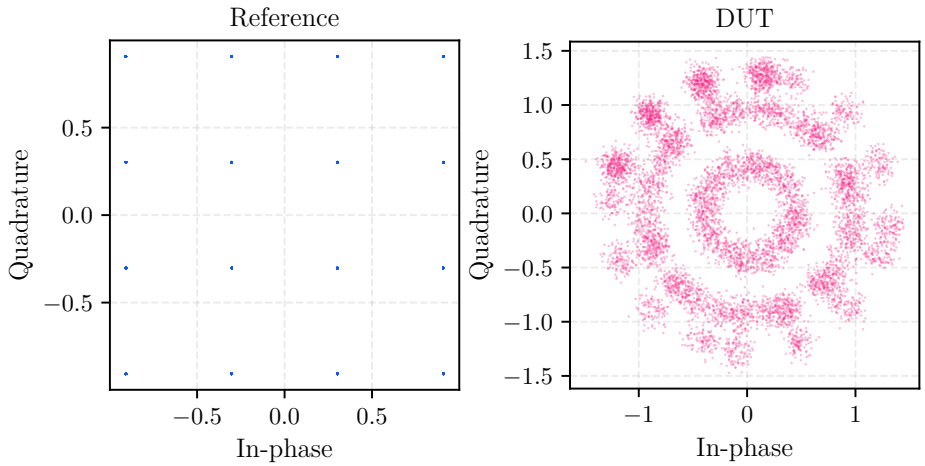

Figure 5.10: Constellation of clean and corrupted data.

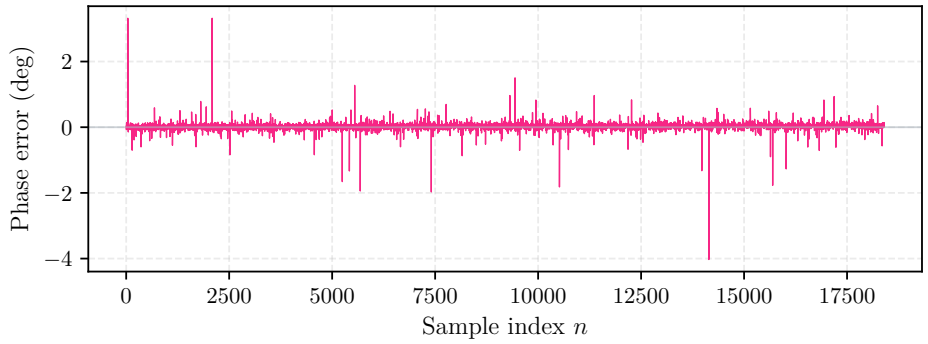

Figure 5.11: Phase error between clean and compensated data.

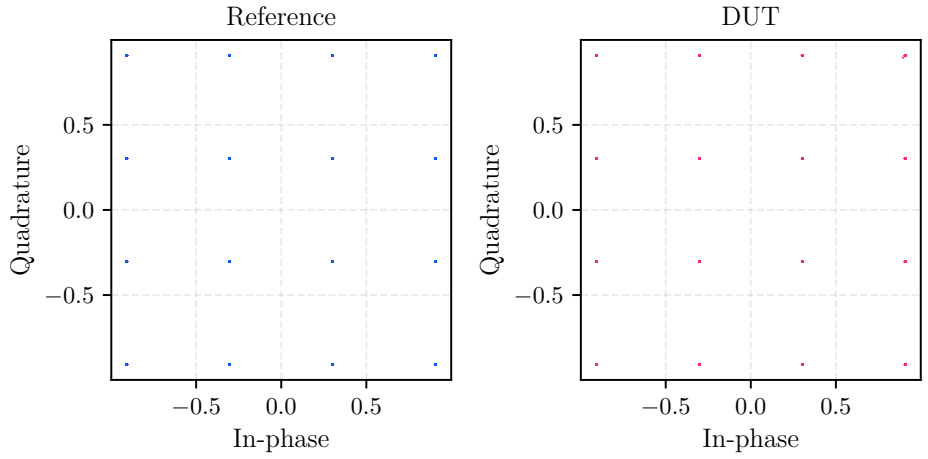

Figure 5.12: Constellation of clean and compensated data.

Fig. 5.11 and 5.12 illustrate the effect of CFO compensation on a single frame, with the first eight symbols shown. The CFO applied to corrupt the signal matches the value derived in Section 2.3.1. The plots show a mostly compensated CFO with some minor artifacts that are attributable to rounding errors inherent to the hardware implementation. A more pronounced anomaly becomes apparent when a longer frame is examined, where a slow but steady phase drift accumulates over time. This effect is discussed in Section 3.3.3, where the numerical discrepancy between the exact value and its fixed point hardware representation is quantified.

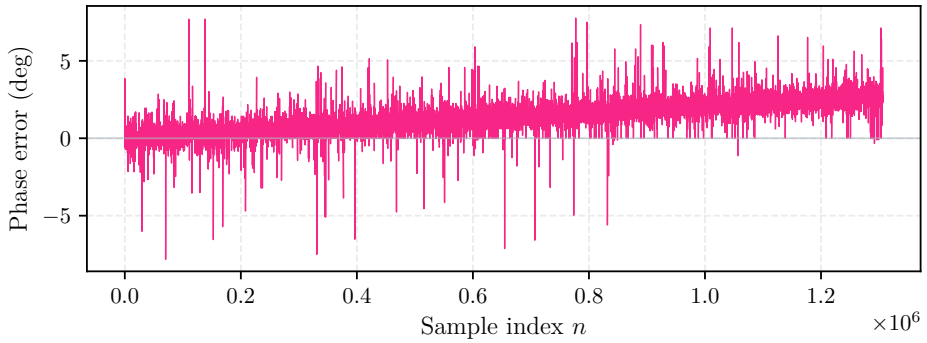

Figure 5.13: Phase drift over a longer frame.

The phase error anomalies of up to a few degrees visible in Fig. 5.11 and Fig. 5.13 are most likely a measurement artifact rather than a compensation failure. Minor artifacts are present even in the comparison between the clean and corrupted signals shown in Fig. 5.9, though they are not visible at that plot scale. These arise from the way the signals are generated and saved in Python. The clean signal samples are rounded to the nearest integer directly, while the corrupted signal samples are produced by multiplying the clean signal by a complex exponential, which generally yields non-integer values that are then rounded independently. The rounding errors for the two signals are therefore uncorrelated and depend on the rotation angle at each sample, introducing small irregularities. These are most pronounced at samples where the signal amplitude is locally low, since a fixed rounding error represents a larger phase deviation at reduced amplitude.

5.3 System Evaluation

Although the primary focus of this work is on the first steps toward a fully bistatic radar, namely synchronization, compensation for the effects of frontend decoupling, and the development of a data structure and beam index pattern compatible with the new data format. The system was also evaluated as a standalone radar.

To assess system performance, range–Doppler plots were generated to verify the

estimated range and velocity of a target. The test was conducted in a forward scatter configuration, with a reflective metal sheet used as the target. The sheet was placed approximately 95 cm from both the TX and RX, with the two frontends separated by approximately 160 cm.

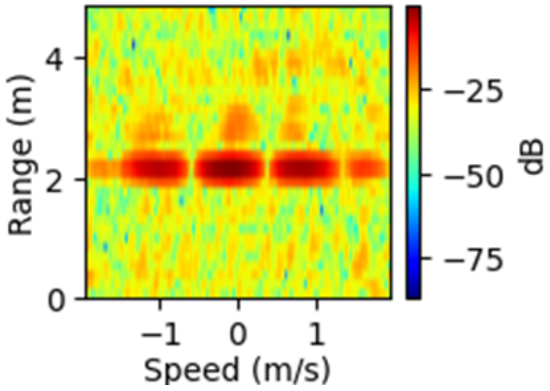

Figure 5.14: Single beam direction range-Doppler map with one target.

As shown in Fig. 5.14, the estimated target range is in good agreement with the known geometry. However, notable artifacts are visible in the range domain. The exact origin of these artifacts has not been fully determined, but several contributing factors are plausible. Residual phase noise, which is not explicitly compensated in the current implementation, can introduce spectral leakage and smearing in the range profile. Additionally, portions of the signal processing chain were originally developed for the monostatic configuration and may not have been fully adapted to the bistatic setup, potentially introducing inconsistencies that manifest as range-domain distortion. More research and testing is needed to isolate the dominant cause.

Placing one target closer to the RX and a second target at a greater distance yields consistent results, with the estimated ranges corresponding to the known target positions. However, noticeable smearing in the range domain persists, consistent with the artifacts observed in the single target case. The range–Doppler plots also exhibit increased noise, attributable to reflections between the two targets.

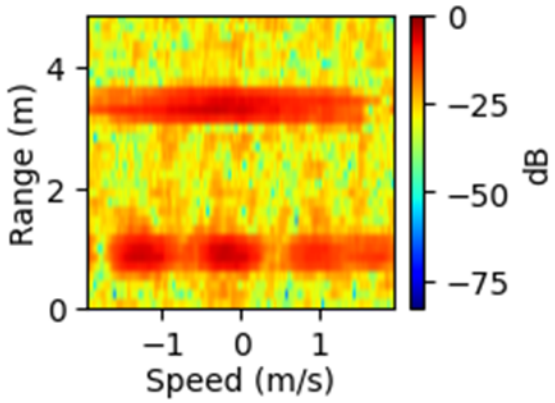

Figure 5.15: Single beam direction range-Doppler map with two targets.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The vision of 6G ISAC rests on the ability of autonomous nodes to operate independently, without shared clocks, cables, or infrastructure. Establishing a reliable link between these nodes requires robust synchronization to withstand the full range of real world impairments, which remains the focus of this work.

The central contribution is the design and implementation of a timing synchronization system based on [16]. Timing synchronization is the prerequisite on which all subsequent processing depends. Without a reliable estimate of frame boundaries, channel estimation, data recovery, and radar processing are all rendered unreliable. The implementation was validated at every stage of the development hierarchy, from an idealized high level Python model, through progressively more constrained fixed point representations, to a full RTL implementation verified against live OTA measurements. The consistency of results across these levels provides confidence in both the chosen algorithms and their implementation.

CFO estimation and compensation were developed as a natural extension of the timing synchronization framework. Rather than introducing an independent estimation mechanism, the approach exploits the correlation metric already computed during timing synchronization to extract frequency offset information. The hardware results confirm that the method performs reliably across the range of CFO values.

The third contribution addresses the structural requirements of a future fully bistatic configuration. Since a bistatic receiver has no direct access to the trans-

mitted waveform, the sensing function must be entirely supported by the received signal. In preparation for this, a new data format and beam indexing scheme were designed and validated in a quasi-bistatic setup, providing the structural foundation on which a fully bistatic extension can be built.

6.2 Limitations

The system operates over a signal bandwidth of 1966.08 MHz, derived from a DAC output rate of 3932.16 MSPS, placing substantial demands on the underlying hardware. As the entire design is implemented on a single FPGA board, on-chip memory resources represent a significant constraint. The baseline monostatic design already consumed a large portion of the available block RAM, leaving limited headroom for the additional components required by the bistatic extension. Consequently, careful and efficient memory management was a persistent design requirement throughout the project. Compromises to OFDM frame length and range-Doppler resolutions had to be incorporated into the design to maintain all signal processing within the confines of the resource requirements.

Since this work represents a first step towards a fully bistatic setup, one of the main limitations encountered was the absence of compensation for SFO and CPO. While timing synchronization and CFO estimation and compensation proved sufficient to yield acceptable radar performance, the residual impairments introduced by uncompensated SFO and CPO made it difficult to isolate the cause of smearing in the range-Doppler maps, which complicated systematic debugging.

6.3 Future work

Several steps remain on the path towards a fully bistatic configuration. The final hardware connection between the TX and RX units must eventually be eliminated, which requires the receiver to reconstruct a local reference independently. This will require the addition of channel estimation and compensation, enabling a reference symbol to be derived from the first transmission of each unique data symbol. Practically, this involves pilot extraction and frequency domain interpolation to recover the channel response across the full subcarrier grid. Given the resource demands of these additions to the already existing signal processing chain, this implementation will most likely need to be distributed across two separate RFSoC FPGAs. The system would need to be partitioned across two boards, with the transmit chain implemented on one board and the more advanced receive chain on the other. One approach to reducing resource utlization is to restrict radar processing to pilot subcarriers [3].

The work also leaves open the question of the origin of the range domain artifacts

observed in the radar measurements. A systematic investigation into their cause is necessary before the sensing performance can be fully characterized. Phase noise compensation, in particular, is a likely candidate for further development, as its absence may be a contributing factor to the observed distortion and could, once addressed, significantly improve the quality of the range–Doppler plots.

6.3.1 Analog Beamforming

An interesting and novel exploration for further work remains in chamber testing of different beamforming patterns. This work used preset beam shapes based on a beam book provided by Sivers. The company provided gain and phase settings for preset beam shapes that were suitable for this current work but some interesting results could manifest from setting new phase and gain values for customized beam shapes. These custom beam shapes could provide more insights on the nature of the side lobe synchronization.

Characterizing the antenna radiations patterns of the EVK06005 frontend units in an anechoic chamber would provide a precise baseline for understanding the true beam shapes produced by the phased array beamformer. Chamber measurements would allow the main lobe width, side lobe levels, and null positions to be accurately mapped across the full range of available beam directions. Beyond characterization, access to direct phase and gain control over individual antenna elements would open the possibility of designing custom beam shapes tailored specifically to the geometry of the fully bistatic radar setup.

References

[1] Y. Wu, A. R. Arifianto, A. A. Gebremariam, V. Unnikrishnan, and L. Liu, "A 28-GHz monostatic OFDM-based ISAC system on RFSoC with real-time processing," in *Proceedings of the 2025 IEEE Nordic Circuits and Systems Conference (NorCAS)*. IEEE, 2025.

[2] Y. Chen, "Radar-based gesture recognition for extended reality applications," Master's Thesis, KTH Royal Institute of Technology, Stockholm, Sweden, October 2025.

[3] D. Brunner, L. Giroto de Oliveira, C. Muth, S. Mandelli, M. Henninger, A. Diewald, Y. Li, M. Basim Alabd, L. Schmalen, T. Zwick, and B. Nuss, "Bistatic ofdm-based isac with over-the-air synchronization: System concept and performance analysis," *IEEE Transactions on Microwave Theory and Techniques*, vol. 73, no. 5, pp. 3016–3029, 2025.

[4] T. Schmidl and D. Cox, "Robust frequency and timing synchronization for ofdm," *IEEE Transactions on Communications*, vol. 45, no. 12, pp. 1613–1621, 1997.

[5] Y. S. Cho, J. Kim, W. Y. Yang, and C. G. Kang, *Introduction to OFDM*. John Wiley Sons, Ltd, 2010, ch. 4.1, p. 111.

[6] R. W. Chang, "Synthesis of band-limited orthogonal signals for multichannel data transmission," *The Bell System Technical Journal*, vol. 45, no. 10, pp. 1775–1796, 1966.

[7] S. Weinstein and P. Ebert, "Data transmission by frequency-division multiplexing using the discrete fourier transform," *IEEE Transactions on Communication Technology*, vol. 19, no. 5, pp. 628–634, 1971.

[8] 3rd Generation Partnership Project (3GPP), "Evolved universal terrestrial radio access (e-utra); lte physical layer; general description," ETSI, 3GPP TS 36.201 19.0.0, 2024.

[9] (3GPP), "Nr; nr and ng-ran overall description; stage 2," ETSI, 3GPP TS 38.300 16.9.0, 2020.

[10] M. Sreekantamurthy, D. C. Popescu, and R. P. Joshi, "Classification of digital modulation schemes in multipath environments using higher order statistics," in *SoutheastCon 2016*, 2016, pp. 1–6.

[11] Y. S. Cho, J. Kim, W. Y. Yang, and C. G. Kang, *Introduction to OFDM*. John Wiley Sons, Ltd, 2010, ch. 4.2, p. 130.

[12] The MathWorks, Inc., "Orthogonal frequency-division multiplexing (OFDM) explained," https://www.mathworks.com/discovery/ofdm.html, 2026, accessed: May 13, 2026.

[13] DSPIllustrations, "Basic OFDM modulation and demodulation," https://dspillustrations.com, accessed: May 13, 2026.

[14] S. G. Li, Ye (Geoffrey), *Orthgonal Frequency Division Multiplexing for Wireless Commuincations*. Springer Science+Business Media, Inc, 2006, ch. 1.5, p. 12.

[15] A. Manikas, "Bistatic radar," Lecture Notes, EE3-27: Principles of Classical and Modern Radar, February 2020, v.21.

[16] H. Minn, M. Zeng, and V. Bhargava, "On timing offset estimation for ofdm systems," *IEEE Communications Letters*, vol. 4, no. 7, pp. 242–244, 2000.

[17] C. Sturm, E. Pancera, T. Zwick, and W. Wiesbeck, "A novel approach to ofdm radar processing," in *2009 IEEE Radar Conference*, 2009, pp. 1–4.

[18] C. Sturm and W. Wiesbeck, "Waveform design and signal processing aspects for fusion of wireless communications and radar sensing," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1236–1259, 2011.

[19] *Synchronization for OFDM*. John Wiley Sons, Ltd, 2010, ch. 5, pp. 153–185. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470825631.ch5

[20] T. E. Abrudan, A. Haghparast, and V. Koivunen, "Time synchronization and ranging in ofdm systems using time-reversal," *IEEE Transactions on Instrumentation and Measurement*, vol. 62, no. 12, pp. 3276–3289, December 2013.

[21] 3rd Generation Partnership Project (3GPP), "Evolved universal terrestrial radio access (e-utra); base station (bs) radio transmission and reception," ETSI, 3GPP TS 36.104 11.18.0, 2021, release 11. [Online].

Available: https://www.etsi.org/deliver/etsi_ts/136100_136199/136104/11.18.00_60/ts_136104v111800p.pdf

[22] A. Manikas, "Phased-array radar," Lecture Notes, EE3-27: Principles of Classical and Modern Radar, February 2020, v.25a.

[23] C. A. Balanis, *Antenna Theory: Analysis and Design*, 4th ed. Hoboken, NJ: John Wiley & Sons, 2016.

[24] H. Minn, V. K. Bhargava, and K. B. Letaief, "A robust timing and frequency synchronization for ofdm systems," *IEEE Transactions on Wireless Communications*, vol. 2, no. 4, pp. 822–839, July 2003.

[25] MathWorks, "Wireless communications toolbox," https://www.mathworks.com/products/wireless.html, 2026, accessed: 2026-06-13.

[26] Python Software Foundation, "Python programming language," https://www.python.org, 2026, accessed: 2026-06-13.

[27] C. R. Harris, K. J. Millman *et al.*, "Array programming with numpy," *Nature*, vol. 585, pp. 357–362, 2020.

[28] W. Lin and C. Huan, "A timing synchronization and frequency offset estimation for ofdm systems based on pn (pseudo-noise) sequence," in *2007 8th International Conference on Electronic Measurement and Instruments*, 2007, pp. 1–652–1–655.

[29] S. Wensheng and Z. Yuanyuan, "A frame synchronization and symbol timing synchronization algorithm in burst ofdm communication based on ieee802.11a," in *2009 International Forum on Information Technology and Applications*, vol. 1, 2009, pp. 190–193.

[30] B. Park, H. Cheon, C. Kang, and D. Hong, "A novel timing estimation method for ofdm systems," *IEEE Communications Letters*, vol. 7, no. 5, pp. 239–241, 2003.

[31] T. Erseghe and L. Vangelista, "Exact analytical expression of schmidl-cox signal detection performance in awgn," *IEEE Communications Letters*, vol. 14, no. 5, pp. 378–380, 2010.

[32] J. Seo, J. Joo, G. Zhu, and S. C. Kim, "Esim ofdm with schmidl and cox algorithm synchronizer in rayleigh fading channel," in *2019 25th Asia-Pacific Conference on Communications (APCC)*, 2019, pp. 267–270.

[33] DSP Illustrations. (2024) Schmidl & cox synchronization for ofdm. [Online]. Available: https://dspillustrations.com/pages/posts/misc/schmidlcox-synchronization-for-ofdm.html

[34] Xilinx, Inc., "PYNQ documentation," https://pynq.readthedocs.io/en/v2.1/index.html, 2018, version 2.1. Accessed: 14-05-2026.

[35] A. De Luca, "Forward scatter radar: Innovative configurations and studies," PhD thesis, University of Birmingham, Birmingham, United Kingdom, February 2018.

Appendix A

Appendix

Larger format diagrams are provided below.

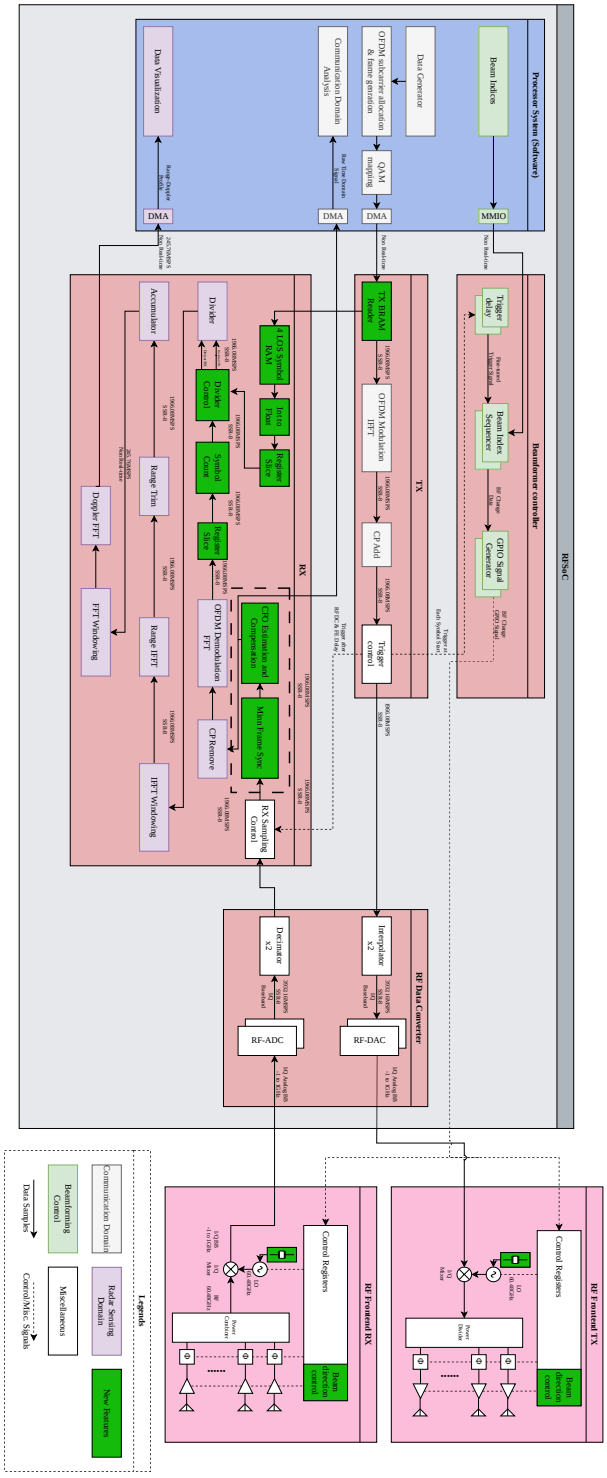

Figure A.1: Large format radar system diagram

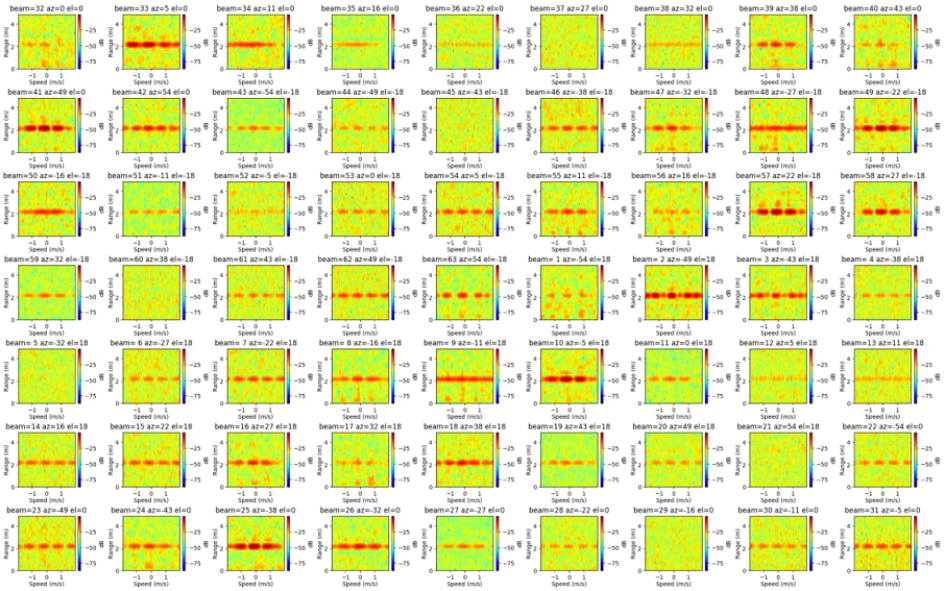

Figure A.2: Full scan range-Doppler map with one target.

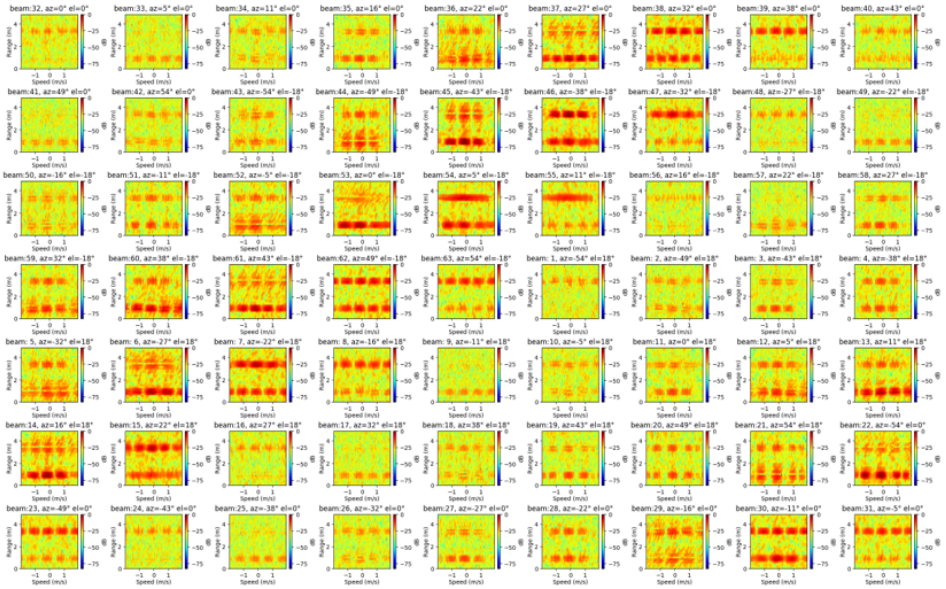

Figure A.3: Full scan range-Doppler map with two targets.