A LOW-MEMORY SPECTRAL-CORRELATION ANALYZER FOR DIGITAL QAM-SRRC WAVEFORMS

DYLAN JACOB GORMLEY

Bachelor of Science in Computer Engineering

University of Maryland, Baltimore County

May 2017

Submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

at the

CLEVELAND STATE UNIVERSITY

MAY 2021

We hereby approve this thesis for

DYLAN JACOB GORMLEY

Candidate for the Master of Science in Electrical Engineering degree for the

Department of Electrical Engineering and Computer Science

and the CLEVELAND STATE UNIVERSITY'S

College of Graduate Studies by

Professor Chansu Yu — Committee Chairperson

Department — Date

Associate Professor Pong P. Chu — Committee Member

Department — Date

Associate Professor Murad Hizlan — Committee Member

Department — Date

Student's Date of Defense

DEDICATION

To my dad, for his unconditional love and support.

ACKNOWLEDGEMENTS

I would like to thank the following people and organizations in no particular order:

My adviser, Professor Murad Hizlan, for all that he has taught me about communications and signal processing theory over the last few years. He has nurtured every step of my journey through his flexibility, patience, and encouragement. Signal detection and estimation theory has changed the way I view the world.

NASA Glenn Research Center for funding my master's degree. Working in the Cognitive Signal Processing branch is a dream come true. Thanks, Gene and Dave! My colleagues, Joe, Mick, Adam, and Aaron, have shaped me into the researcher I am today. This team is the most cohesive, warm, and driven group of people that I have ever worked with. They even took the time to help me with my homework for this degree– Impeccable. I hope there are many years to come for the R119 crew. Time to put this thing in space.

Doctor Chad M. Spooner for his feedback on the theory in this thesis. He kept me accountable on every equation, every plot, and every concept. Without his cyclostationary blog, I doubt this thesis would have been possible.

Anthony A. Stock for his independent verification and validation of the digital designs seen in this paper. I have enjoyed watching him grow into a competent engineer over the last year. He has worked so incredibly hard, and I am proud of him. I would like to wish him good luck with whatever comes next in his life!

Erica K. Henrichsen for creating the block diagrams seen in this paper. Besides her amazing skills as a graphic designer, she was the one that helped me get through the day-to-day trials of life. I would like thank her for believing in me, even when I don't.

My dad, John; my step-mom, Barb; my mom, Amy; and my sister, Cailin for tolerating me over the last few decades– I love them all very much. My cousins

Megan, Mandi, Jimmy, and Liz, for picking me up during my darkest days, year after year after year. My friends David, Greg, Jake, and Kent for reminding me that life can be fun– They are like brothers to me. Lastly, and most importantly, my cat Copernicus for keeping me company during the arduous process of writing this thesis.

A LOW-MEMORY SPECTRAL-CORRELATION ANALYZER FOR DIGITAL QAM-SRRC WAVEFORMS

DYLAN JACOB GORMLEY

ABSTRACT

Cyclostationary signal processing (CSP) provides the ability to estimate received waveforms' statistical features blindly. Quadrature amplitude modulated (QAM) waveforms, when filtered by the square-root-raised cosine (SRRC) pulse shape function, have cyclic features that CSP can exploit to detect waveform parameters such as symbol rate (SR) and center frequency (CF). The estimation of these SR-CF pairs enables a cognitive radio (CR) to perform spectrum sensing techniques such as spectrum sharing and interference mitigation. Here, we investigate a field-programmable gate array (FPGA) application of a blind symbol rate-center frequency estimator. First, this study provides a background on the theory behind the cyclic spectral density function (CSD), spectral correlation analyzers (SCA), and spectrum sensing. Following this is a discussion on the motivation for CubeSat spectrum sensing. An SCA implementation for low-memory devices, such as FPGA-based CubeSat, is then describes. The paper concludes by reporting the performance characteristics of the newly developed streaming-based SCA.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I
INTRODUCTION

Modulated radio-frequency (RF) communication signals exhibit probabilistic parameters that vary periodically with time. These statistical properties are known as *cyclostationarity*. This thesis is interested in estimating the cycle frequencies of second-order cyclostationary (SOC) signals. In this work, we perform temporal analysis using the cyclic autocorrelation function (CAF). Further, we perform spectral analysis using the cyclic spectral density function (CSD) [1].

Some of the standard CSD estimation techniques are the frequency-smoothing method (FSM), time-smoothing method (TSM), strip spectral correlation analyzer (SSCA), and the fast Fourier transform (FFT) accumulation method (FAM) [1]. The realization of a CSD estimator is called a *spectral correlation analyzer* (SCA) [1][2]. This paper investigates SCA techniques and evaluates their ability to be used with severely resource-constrained platforms, such as cube satellites (CubeSats).

Terrestrial receivers have few constraints to consider with regard to size, weight, and power (SWaP). High-SWaP systems such as multi-core central processing units (CPU), graphics processing units (GPU), and large-cell-count field-programmable gate arrays (FPGA) are deployed to perform complex digital signal processing (DSP) routines.

For CubeSat cognitive radios (CR), however, resources are scarce. These CRs are tasked to collect data, process signals, receive commands, report status, process overhead, and store data, among multiple other advanced functions. All of this functionality must operate under severe SWaP constraints. Hence, the utilization of low-SWaP devices such as low-cell-count FPGAs are deployed to meet these constraints.

In the current state-of-the-art, SCA algorithms utilize large amounts of block random access memory (BRAM) due to the extensive use of the FFT. This paper studies an SCA algorithm's design with a low reliance on BRAM by processing in the time domain. The low-BRAM design operates entirely on a sample-by-sample (”streaming”) basis, thus significantly reducing the BRAM required. As a result of the decreased need for BRAM, a streaming-based SCA can be deployed onto CubeSat CRs.

Chapter II begins by reviewing the basic notation and concepts of stochastic processes, also known as *random signals*. We then develop the intuition and mathematics behind cyclostationary signal processing (CSP). Following, we discuss the challenges associated with calculating the power spectral density (PSD) of random signals. PSD estimation techniques are discussed, followed by CSD estimation techniques for cyclostationary signals. After this, we evaluate several SCAs and discuss their appropriateness for our low-BRAM objective. Finally, we conclude with an overview of a technique that leverages CSP to intelligently reused spectral bands, known as *spectrum sensing*.

In Chapter III, we first provide commentary on the motivation for performing spectrum sensing in space. Following, we discuss a set of assumptions necessary for the design of our low-BRAM SCA. We model a previously studied SCA, which we will refer to as the CAF - discrete Fourier transform (DFT) (CAF-DFT) algorithm, in matrix laboratory (MATLAB) to validate its use on M-order quadrature amplitude modulated waveforms (QAM) with square-root-raised-cosine (SRRC) pulse-shaping

(M-QAM-SRRC). Due to CubeSat CR limitations, we simplify parts of the design using a single-point estimator with little BRAM. We conclude this chapter with comprehensive look into the digital architecture for such a design, which we will refer to as the *streaming-SCA*, due to its BRAM-reducing stream-based approach.

In Chapter IV, we perform a characterization of our newly created streaming-SCA. Experiments will stress-test various parameters, including bit widths, symbol rates, center frequencies, capture lengths, noise, and SRRC roll-offs. We conclude this thesis in Chapter V with an overall summary and a discussion of opportunities for future work.

CHAPTER II
BACKGROUND

In this chapter, we review the concept of a random signal. We then briefly discuss the autocorrelation function (AF), followed by the CAF. After this, we describe the challenges of PSD and CSD estimation. Finally, we conclude with an overview of a technique that leverages CSP to reuse free spectral bands, known as *spectrum sensing*.

2.1 Cyclostationary Signal Processing

In this section, we summarize random signal notation and concepts as described in W. A. Gardner's book *Introduction to Random Processes with Applications to Signals and Systems* [3]. We begin with the AF and then build on this groundwork to introduce the concept of the CAF.

2.1.1 The Autocorrelation Function

We begin with a review of stochastic processes. For digital communications these are typically referred to as *random signals*. The random signal $X(t)$ is an infinite collection of individual signals plus probability density functions (PDF). A member of $X(t)$ is the sample path or time-series, denoted as $x(t)$. We will declare $X(t)$ a complex-valued random signal such that $X(t) \sim \mathcal{N}(t; \mu_X(t), \sigma_X(t))$, where $\mathcal{N}$ is the Gaussian PDF of the form:

$$p(d) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{d-\mu}{\sigma}\right)^2} \tag{1}$$

Moreover, it has the shape shown in Figure 1.

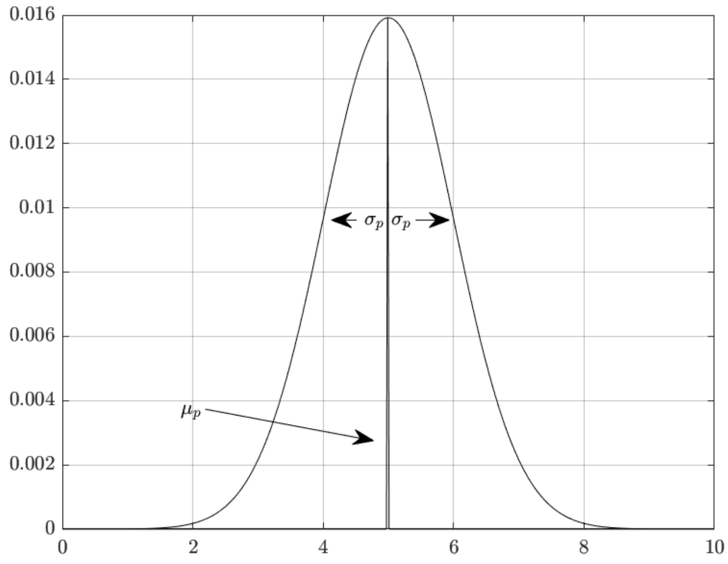


Figure 1: Example of a Gaussian PDF. Here, Equation (1) is evaluated for 500 values of d, with a mean of $\mu = 5$, and a standard deviation of $\sigma = 1$.

We generally characterize a random signal by specifying all n^{th}-order joint PDFs for its values, known as *probabilistic parameters*. To properly define a signal's probabilistic parameters, we start with the expected value $E_p\{X(t)\}$. If $X(t)$ is only a function of time, we find that we can model the PDFs under a framework known as *fraction-of-time probability* [1]. Assuming $X(t)$ is modeled under the fraction-of-time framework $E\{X(t)\}$ is defined as

$$E_p\{X(t)\} = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} X(t,s)p(s)ds \tag{2}$$

Using Equation (2), we can determine probabilistic functions, known as *moments*. The first moment, defined as $E_p\{X(t)\}$, is known as the *mean* of $X(t)$, denoted $\mu_X(t)$. Assuming no skew, the mean is equivalent to the expected value for a Gaussian random signal [3].

The second moment, defined as $E_p\{(X(t) - \mu_X(t))^2\}$, is known as the *variance* of $X(t)$, denoted $\sigma_X^2(t)$. The variance is commonly expressed as its square-root, known as the *standard deviation* $\sigma_X(t)$. The standard deviation is used to express the the spread of values for $X(t)$ [3].

The second joint moment of $X(t)$ with itself is defined as $E_p\{X(t_1)X^*(t_2)\}$ at times (t_1, t_2), and is known as the *AF*, denoted $R_X(t_1, t_2)$. For WSS random signals, the AF is expressed as $R_X(\tau)$, where $\tau = t_1 - t_2$. Intuitively, variable τ is the amount of time that t_2 lags behind t_1 and is thus typically referred to as the *lag variable* [3].

If $X(t)$ exhibits time-invariance of its probabilistic parameter functions, then it is said to be *stationary.* If all probability densities up to order n are time-invariant, the process is said to be n^{th}-order stationary. A process that is n^{th}-order stationary for all possible values of n is said to be *strict-sense stationary.* If only the mean function and AF are time-invariant, the process is said to be 2^{nd}-order stationary (SOS) or *wide-sense stationary* (WSS) [1][3].

Let us further develop $X(t)$ by assuming it is a WSS random signal. The WSS AF of $X(t)$ is defined as,

$$R_X(t, \tau) = \lim_{T \to \infty} \frac{1}{T} \int_{-T/2}^{T/2} X\left(t + \frac{\tau}{2}\right) X^*\left(t - \frac{\tau}{2}\right) \, dt \tag{3}$$

For a WSS AF, the mean function and standard deviation function values are the same for all possible points in time [3], as illustrated in Figure 2.

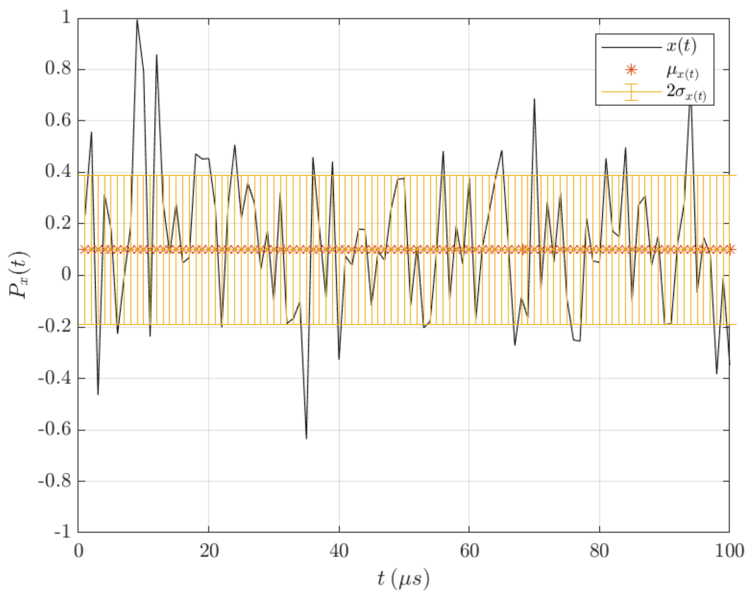


Figure 2: Example of a WSS Gaussian random signal. Here, we observe Equation (1) for 100 points in time, with $X(t) \sim \mathcal{N}(t; \mu_X(t), \sigma_X(t))$ and $\mu_{x(t)} = 0.1$, $\sigma_{x(t)} = 0.25$ for all $x(t)$.

Most real-world random signals– such as digital communications waveforms– are not WSS. In this context, we will define *waveform* as any modulated signal. One such digital waveform, known as *QAM*, transmits information by modulating the amplitude and phase of two carrier signals creating what is known as *symbols*. The number of combinations possible for a QAM waveform is defined by its *modulation order* (M), denoted M-QAM. The simplest case of M-QAM is $M = 2$, where the waveform alternates between two phases, 180° apart. The rate at which these symbols alternate is known as the waveform's *symbol rate* (SR), denoted R_s. The modulated carrier frequency is known as the waveforms *center frequency* (CF), denoted f_c. An example of an arbitrary 2-QAM time series is shown in Figure 3.

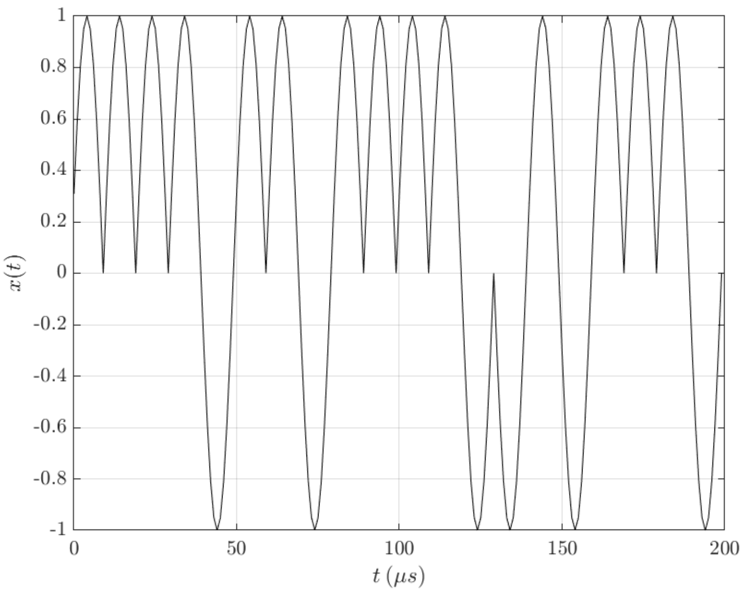


Figure 3: Example of a 2-QAM waveform $x(t)$. Here, we modulate 200 random bits, followed by rectangular pulse-shaping of length 10. Our waveforms parameters are $R_s = 0.1MBd$ and $f_c = 0.05MHz$.

An M-QAM waveform has the mathematical representation,

$$X(t) = \sum_{k=-\infty}^{\infty} a_k p(t - kT_0 - t_0) e^{j2\pi f_0 t + j\theta_0} \tag{4}$$

where a_k is the k^{th} symbol, $p(t)$ is the pulse-shaping filter (PSF), and f_0 is the CF. This definition also covers the general representation of phase-shift keying (PSK) and pulse modulation (PM) waveforms. These M-QAM-SRRC waveforms are of keen interest to communication systems that use link adaptation, such as DVB-S2, WiMAX, HSDPA, and UMTS. Thus for this thesis, we will narrow the scope of our investigation to the M-QAM-SRRC class of waveforms.

To conclude this section, Figure 4 plots $R_X(\tau)$ versus τ as what is known as an *autocorellogram.*

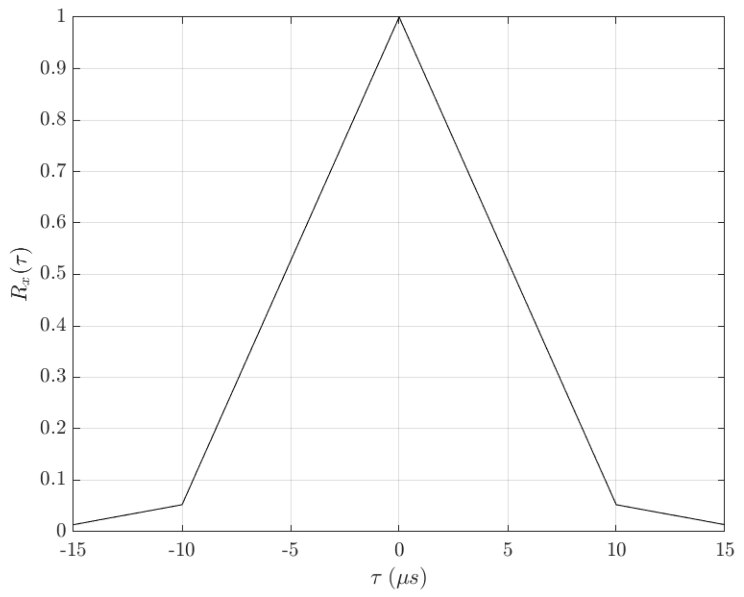


Figure 4: Autocorrelogram of the example waveform $x(t)$ shown in Figure 3. Here, we evaluate Equation (3) for 31 evenly-spaced values of τ.

2.1.2 The Cyclic Autocorrelation Function

If the probabilistic parameters of $X(t)$ vary periodically with time, $X(t)$ is said to exhibit *cyclostationarity*. If only the mean and autocorrelation vary periodically, it is said to be *SOC* or *wide-sense cyclostationary* (WSC) [1][3]. The WSC CAF for $X(t)$ is defined as

$$\widetilde{R}_X^\alpha(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} R_X(t,\tau) e^{-j2\pi\alpha t} dt \tag{5}$$

Equation (5) can be thought of as the cross-correlation between the AF $R_X(t,\tau)$ and the periodic sinusoid $e^{-j2\pi\alpha t}$, where α is called the *cycle frequency parameter*. The expected value of a cyclostationary process is denoted as $E^\alpha\{X(t)\}$. Further, cycloergodicity is a random signal property that is used to relate $E\{X(t)\}$ to $E^\alpha\{X(t)\}$ when we wish to relate fraction-of-time probability to conventional random-process ensemble probability [1]. These assumptions together create the cycloergodic WSC

CAF of $X(t)$. The equation for such a function is written as,

$$R_X^\alpha(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} X\left(t + \frac{\tau}{2}\right) X^*\left(t - \frac{\tau}{2}\right) e^{-j2\pi\alpha t} dt \tag{6}$$

Figure 5 plots the Equation (6) as what is known as the cyclic autocorrelogram.

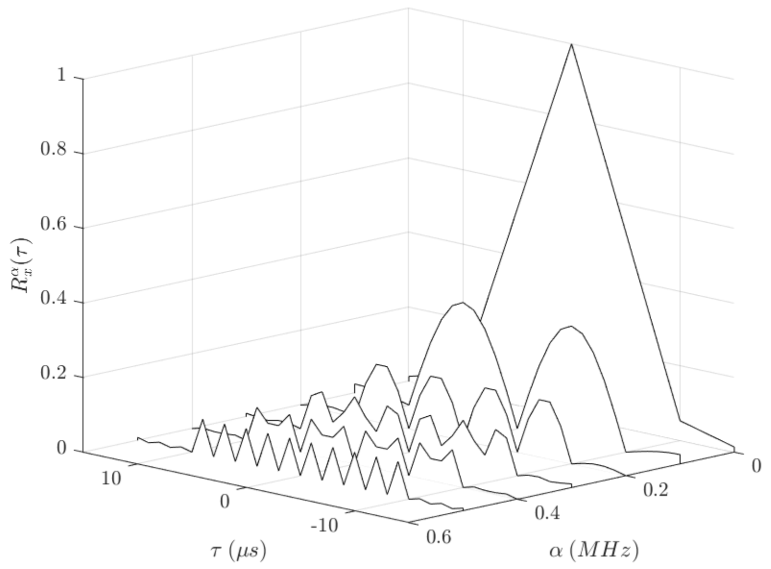


Figure 5: Cyclic autocorrelogram of the waveform $x(t)$ shown in Figure 3. Here, we evaluate Equation (6) for 31 evenly-spaced values of τ and 6 evenly spaced values of α.

Each of the (α, τ) peaks in Figure 5 is a cycle feature of $X(t)$, telling us more information than the AF can alone. We will build on this concept in the following three sections to explain the significance of cyclic features.

2.2 Spectral Density

This section will continue to the summary of notation and concepts described in [3]. We first review the periodogram, the challenges faced when estimating the PSD from the periodogram, and a PSD estimation method known as *frequency-smoothing*. Then we will build upon this work to develop the concept of the CSD.

2.2.1 The Power Spectral Density Function

A rectangular pulse is defined as

$$\Pi(t) = \begin{cases} 0, & if \quad |t| > \frac{1}{2} \\ 1, & if \quad |t| \leq \frac{1}{2} \end{cases} \tag{7}$$

The Fourier transform of $\Pi(t)$ is denoted as $\hat{\Pi}(f)$. Since $\Pi(t)$ is a finite-energy signal, we can measure the distribution of the signal's energy over frequency using Parseval's theorem:

$$\Psi_{\Pi}(f) = \left|\hat{\Pi}(f)\right|^2 \tag{8}$$

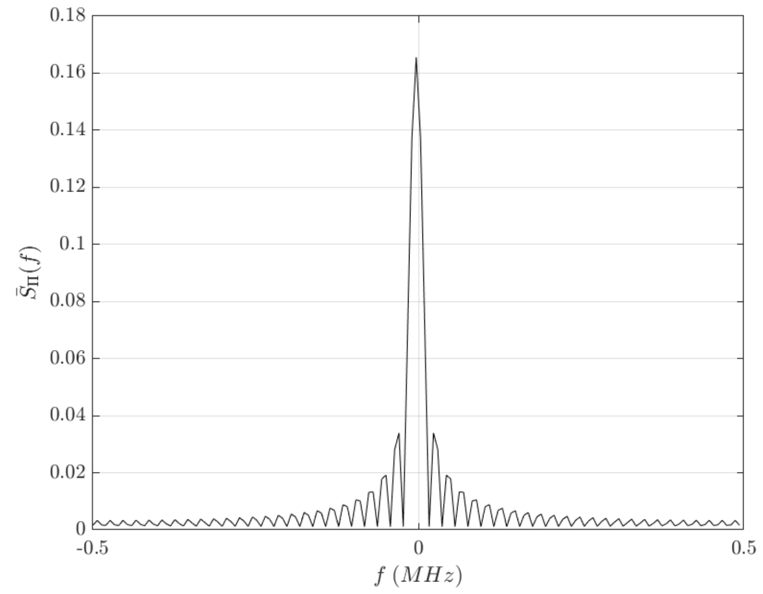


Figure 6: Example of an ESD plot. Here, we evaluate the the Fourier transform of the finite-energy signal $\Pi(t)$ for 150 time instances, resulting in the spectral components expressed as Ψ_{Π}.

Equation (8) is known as the *energy spectral density* (ESD) function [3].

Our QAM waveform $X(t)$ is an random signal and as such does not meet the criterion to perform a Fourier transform, and therefore an ESD does not exist [3]. However, by time limiting $X(t)$ with window size T, the periodogram of $X(t)$ can be expressed as:

$$P_X(f) = \frac{1}{T}\left|\hat{X}_T(t,f)\right|^2 \tag{9}$$

where the Fourier transform of $X(t)$ is denoted as $\hat{X}(f)$. Equation (9) measures the distribution of power over frequency. As a result of time-variance, $P_X(f)$ is an extremely poor estimate with low signal-to-noise ratios [3]. Figure 7 illustrates the periodogram of our 2-QAM signal from 3.

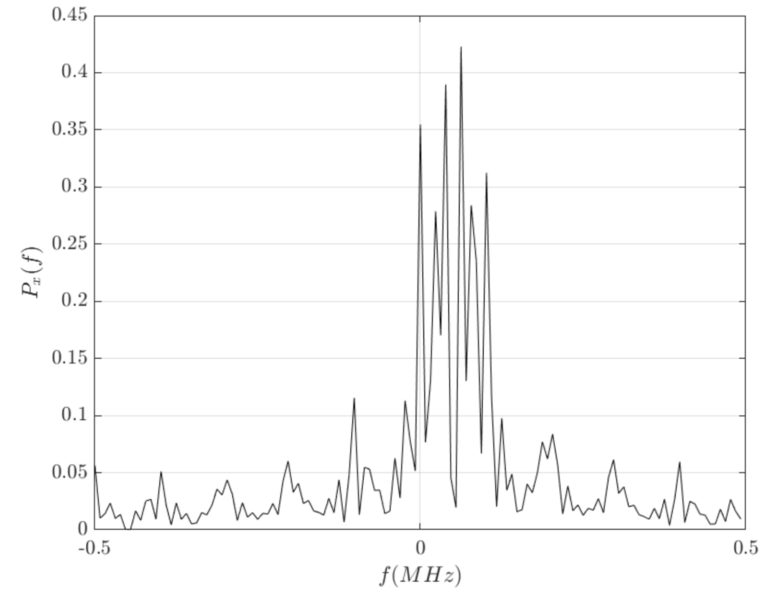


Figure 7: Periodogram of the waveform $X(t)$ shown in Figure 3. Here, we perform an FFT of length 128.

Notice that the periodogram is roughly centered at f_c and roughly the width of two times the R_S, as expected. However, without previous knowledge, it would be challenging to determine the true CF and SR. Thus, we must find a way to improve upon the periodogram to determine our signal's parameters accurately.

One of the techniques used to mitigate this erratic behavior is the FSM [1]. A rectangular smoothing window $h(t)$ is applied to filter $P_X(f)$, which results in the estimate:

$$S_X(f) = h(t) * P_X(t, f) \tag{10}$$

where $*$ is the convolution operator. The frequency-smoothed periodogram has much higher determinism and more closely resembles the actual PSD. Notice in Figure 8 we can estimate the true CF with precision. Smoothing also helps us find the actual SR, however smoothing also widens $S_X(f)$, creating difficulties in detecting the true SR. Therefore, to maximize accuracy a minimal amount of smoothing should always be applied.

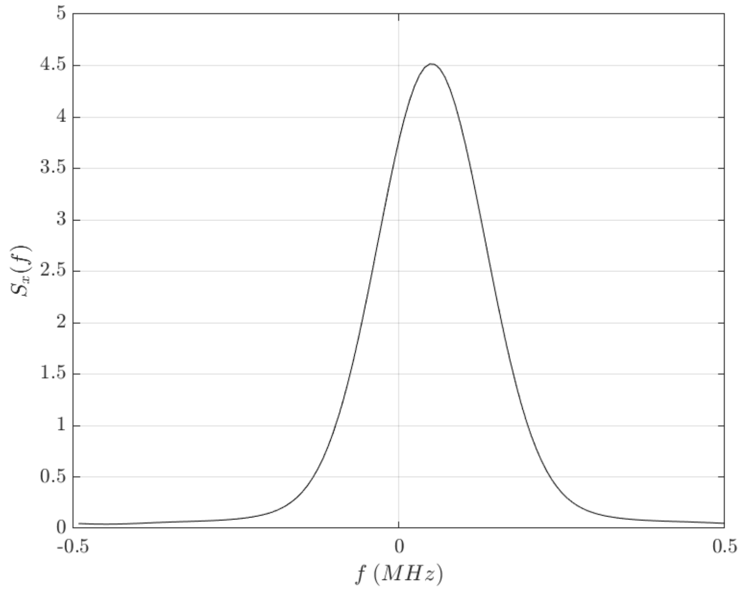


Figure 8: PSD of the waveform $x(t)$ shown in Figure 3. Here, we perform a 128-point FFT of $x(t)$ and a apply a smoothing window of length 8.

2.2.2 The Cyclic Spectral Density Function

In the previous subsection, we found that a random signal's PSD is obtained by applying Parseval's theorem with some smoothing technique to force convergence.

Using this same concept, we can obtain what is known as the *cyclic periodogram*:

$$P_X^{\alpha}(f) = \frac{1}{T}\hat{X}_T\Big(f + \frac{\alpha}{2}\Big)\hat{X}_T^*\Big(f - \frac{\alpha}{2}\Big) \tag{11}$$

where $\hat{X}^*$ is the conjugate of $\hat{X}$. Its corresponding CSD estimate expressed as:

$$S_X^{\alpha}(f) = h(t) * P_X^{\alpha}(t, f) \tag{12}$$

Equation (12) completely characterizes $X(t)$ for all values of cycle frequency α and frequency f [3]. The CSD estimate of $X(t)$ is shown in Figure 9.

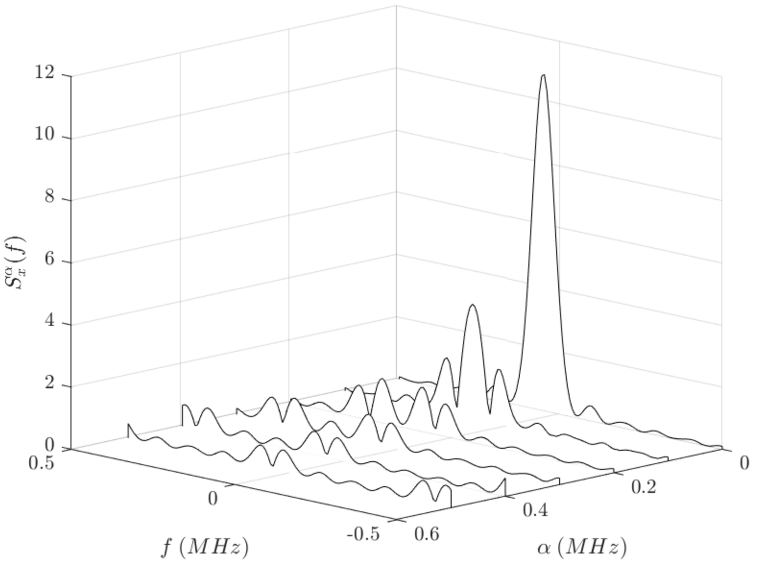


Figure 9: CSD of the waveform $x(t)$ shown in Figure 3. Here, we perform a 128-point FFT of $R_x^{\alpha}(\tau)$ and a apply a smoothing window of length 8.

For $\alpha = 0$, the result is simply the PSD and is always expected to produce a strong result. For $\alpha \neq 0$, patterns can be exploited to reveal statistical features of the received waveform.

2.3 Spectral Correlation Analyzers

In this section, we move onto realizations of the CSD. The process of producing the CSD of received waveform with no previous information is called *blind estimation.* Realizations of a blind estimator are often called *SCAs.* First, we will discuss an example of TSM-SCA as described by W. A. Gardner in his article, *Cyclostationary: Half a Century of Research* [2]. This TSM-SCA is used to blindly estimate the CSD of arbitrary (α, f) pairs.

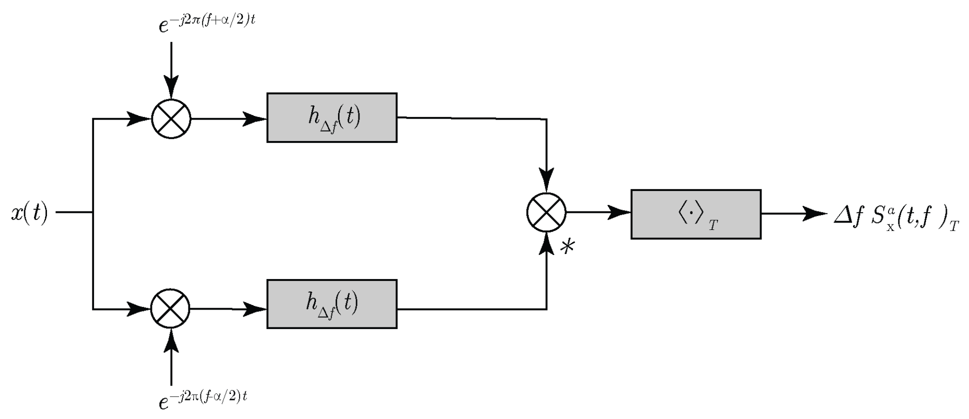
We then discuss the case where the α we wish to estimate blindly is equal to SR of interest. M. V. Koch describes an SCA technique to accomplish this in his technical report, *Interference Mitigation Using Cyclic Autocorrelation and Multi-Objective Optimization* [4], which he refers to as the *CAC-FFT algorithm.* For consistency with our definitions and notations, we will refer to the CAC-FFT as the *CAF-DFT* in this thesis. Using the CAF-DFT, we can blindly estimate SR-CF pairs for M-QAM-SRRC waveforms by letting $\alpha = R_s$ and $f = f_c$.

2.3.1 Blind Cycle Frequency-Frequency Estimation

The most basic realization of the time-smoothed CSD is the TSM-SCA. Rewriting Equation (12) as,

$$S_X^{\alpha}(t,f)_T = \frac{1}{T}\int_{t-T/2}^{t+T/2} \Delta f \hat{X}_T\Big(u, f+\frac{\alpha}{2}\Big)\hat{X}_T^{*}\Big(u, f-\frac{\alpha}{2}\Big)du \qquad (13)$$

We find that the TSM-SCA can decompose into simple multipliers, mixers, filters, and accumulators. To establish a baseline implementation, the block diagram of the TSM-SCA, based on Equation (13), is diagrammed in Figure 10 [2].



$e^{-j2\pi(f+\alpha/2)t}$
$h_{\Delta f}(t)$
$x(t)$
$\langle \cdot \rangle_T$
$\Delta f\, S_x^{\alpha}(t,f)_T$
$*$
$h_{\Delta f}(t)$
$e^{-j2\pi(f-\alpha/2)t}$

Figure 10: As described by Gardner in [2], this baseline SCA is realized by frequency shifting the signal $x(t)$ by two amounts differing by $\alpha/2$, passing such frequency-shifted versions through two low-pass filters $h_{\Delta f}(t)$ with bandwidth Δf and unity pass-band height, and then correlating the output signals. The CSD $S_X^{\alpha}(f)$ is obtained by normalizing the output by Δf, and taking the limit as the correlation time $T \to \infty$ and the bandwidth $\Delta f \to 0$.

While Figure 10 works for any arbitrary waveform and gives the blind CSD estimate for any (α, f) pair, much of this search space is not helpful for most applications. If we can intelligently limit this search space, we can significantly improve the utility of the TSM-SCA.

2.3.2 Blind Symbol Rate - Center Frequency Point Estimation

Several common SCAs such as the FSM, TSM, SSCA, and FAM are discussed in [1]. While these techniques provide speed, they are FFT dependent and therefore memory dependent. Although high-memory allocation is acceptable for terrestrial receivers, we must prioritize memory conservation for a CubeSat CR and explore alternative SCA designs.

In [4], M. A. Koch proposed a modified realization of the SCA, which we will refer to as the *CAF-DFT algorithm.* Koch reports that if we limit our random signal to only M-QAM-SRRC waveforms, then the CSD peaks are always equal to the waveform's actual R_s and f_c [4]. This is proven in [3] and will be summarized below. The discrete AF of Equation (4) is expressed as:

$$R_X(t,\tau) = \sigma^2 \sum_{k=-\infty}^{\infty} p(t+\tau-kT_0)p^*(t-kT_0) \tag{14}$$

Because Equation (14) is periodic with T_0, we can express the CAF as:

$$R_X^\alpha(\tau) = \frac{\sigma^2}{T_0} p(\tau) * \left(p^*(-\tau)e^{j2\pi n/T_0}\right) \tag{15}$$

And the CSD as:

$$S_X^\alpha(f) = \frac{\sigma^2}{T_0} P(f)P^*(f - \frac{n}{T_0}) \tag{16}$$

Lastly, if $P(f)$ is an SRRC PSF, then only $\alpha = -Rs, 0, Rs$ have non-zero cyclic frequencies. Since $\alpha = 0$ is the trivial case, and we do not wish to evaluate the negative symbol rate $\alpha = R_s = -1$, we are left with $S_X^{R_s}(f)$ as the only spectral density with nonzero points. Evaluating the CSD slice of $S_X^{R_s}(f)$, we find that peak is at exactly $f = f_c$, perfectly matching the true (R_s, f_c) of $X(t)$. This finding is illustrated in Figure 11.

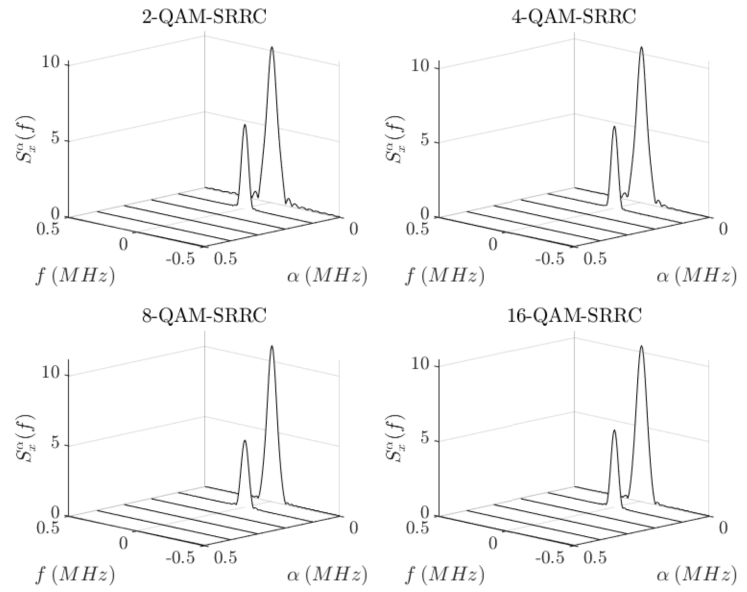


Figure 11: CSD of the waveform $x(t)$ shown in Figure 3, with various values of M. The PSD for $\alpha = 0$ is the spectral density of the conventional AF, which will always return a strong correlation. Barring this trivial case, we notice a strong correlation for $\alpha = 0.1MBd$, which is the same as atual R_s of $x(t)$'s. Additionally, the peak of the PSD for $\alpha = 0.1MBd$ is equal to actual f_c of $x(t)$. Therefore, by finding the nontrivial (α, f) peaks of the CSD for a M-QAM-SRRC waveform, we also find the waveform's actual (R_s, f_c), even with no previous information.

The discrete CAF-DFT is written as,

$$S_X^v[k] = \frac{1}{N}\sum_{n=0}^{N-1} \left|(X[n]e^{-j2\pi kn}) * h[n]\right|^2 e^{-j2\pi vn} \tag{17}$$

Where $v = \lceil \frac{R_s}{fs} N \rceil + 1$, $k = \lceil \frac{f_c}{fs} N \rceil + 1$, and $n = \frac{t}{T}$. The realization of Equation (17) is shown in Figure 12 [4].

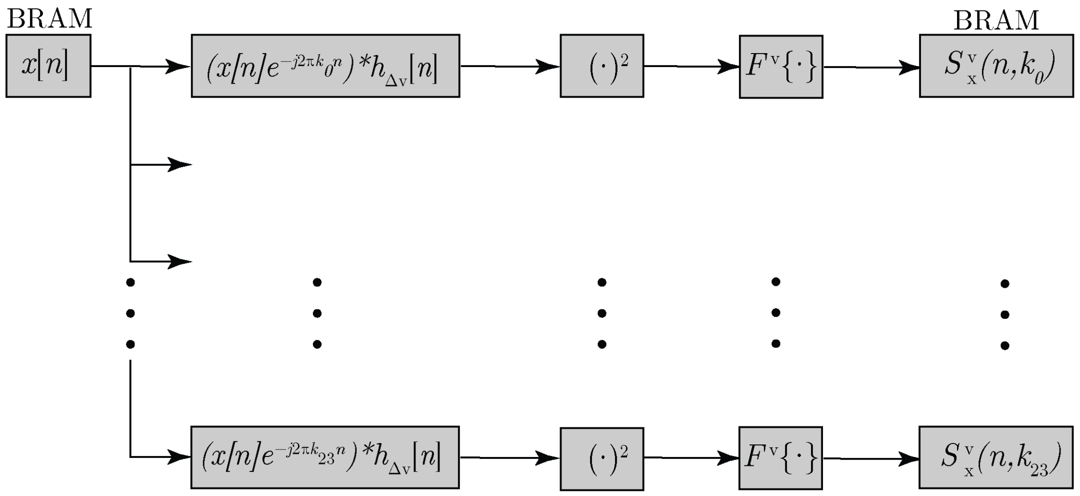


Figure 12: This SCA is realized by frequency-shifting the waveform $x[n]$ by various candidate ks in parallel, passing each frequency-shifted version through a LPF $h_{\Delta v}[n]$ with bandwidth Δv and unity-passband height, in a process referred to as channelization. The autocorrelation of the channelized signal is then performed at $\tau = 0$, which simplifies as the magnitude squared. The DFT of is then performed using a kernel value of v. The CSD $S_X^v[k]$ is obtained by normalizing the output by $1/N$.

As shown in Figure 12, the CAC-DFT has a narrowed search space to estimate only the SR-CF of M-QAM-SRRC waveforms. This assumption uses a constant value of $\tau = 0$, thus avoiding the need to be processed in the frequency domain. Therefore, the CAC-DFT lends itself well to time-domain processing.

2.4 Spectrum Sensing

Spectrum sensing is the process of periodically monitoring a specific frequency band to infer the presence or absence of users. Monitoring is typically performed by a secondary user (SU) to reuse inactive frequencies allocated/licensed to a primary user (PU) [5][6]. These allocated, inactive areas of a band are referred to as *spectral holes* [5]. In this section, we discuss several detection techniques and the strengths of CSP for spectrum sensing.

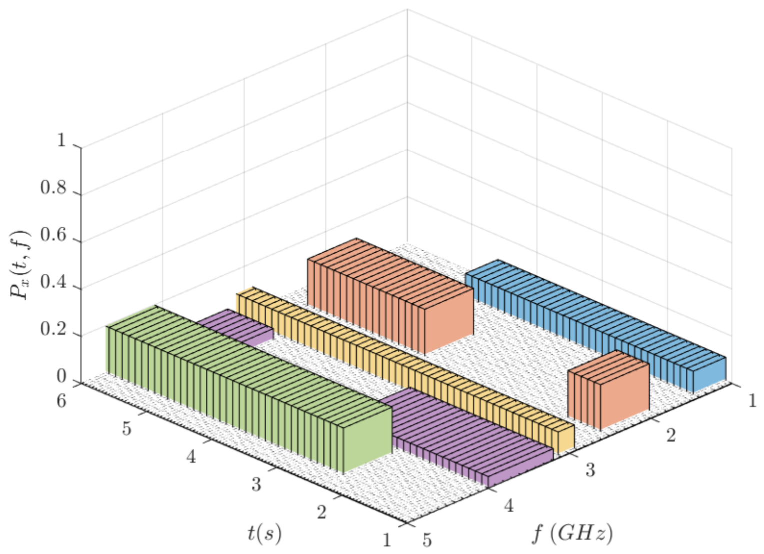


Figure 13: Example of an RF spectrum over time. PUs occupy various bands according to their allocation. However, they may not always need to utilize a band at every moment in time. These gaps in the spectrum use are known as spectral holes. If a SU can maintain awareness of active PUs, the SU can frequency hop between the holes and dynamically change it is SR based on the current hole's allotted bandwidth.

2.4.1 The Binary Hypothesis

Let us consider a SU that is monitoring a narrow RF band for the presence of a PU. There are two hypotheses: H_0, the PU is inactive and H_1, the PU is active [6]. The received signal is written as,

$$H_0 : Y(t) = \eta \tag{18}$$

$$H_1 : Y(t) = G(t) * X(t) + \eta \tag{19}$$

where $X(t)$ denotes the transmitted signal from the primary user, $G(t)$ is the channel response, and η is independent and identically distributed (IID) additive Gaussian white noise (AWGN). According to the binary hypothesis problem, we know that either H_0 or H_1 is true. Our objective is to correctly choose H_0 or H_1, based on the samples received [6]. For every observation we shall define the likelihood of four possible outcomes:

1. Probability of Correct Rejection $P_{cr} = P(H_0|H_0)$: H_0 is true; choose H_0.
2. Probability of Miss $P_m = P(H_0|H_1)$: H_1 is true; choose H_0.
3. Probability of False Alarm $P_{fa} = P(H_1|H_0)$: H_0 is true; choose H_1.
4. Probability of Detection $P_d = P(H_1|H_1)$: H_1 is true; choose H_1.

Outcomes P_d and P_{cr} correspond to correct choices, while outcomes P_m and P_{fa} correspond to erroneous choices. Each of these four probabilities have a corresponding physical meaning in the context of spectrum sensing:

1. P_{cr} defines the likelihood that the SU will transmit.
2. P_m defines the likelihood that the SU will interfere with the PU.
3. P_{fa} defines the likelihood that the SU will miss an opportunity to transmit.
4. P_d defines the likelihood that the SU will not interfere with the PU.

In general, a detection method is said to be *optimal* if it achieves the lowest P_{fa} for a given P_d. In many physical situations, it is not easy to know the probabilities in advance. The Neyman-Pearson theorem [6][7] states that we can constrain P_{fa} to maximize the probability of detection with a likelihood ratio defined as

$$\Lambda(Y(t)) = \frac{P_{Y|H_1}(Y(t)|H_1)}{P_{Y|H_0}(Y(t)|H_0)} \tag{20}$$

We can compare $\Lambda(Y(t))$ to a threshold γ to form a likelihood ratio test [6][7] denoted as

$$\Lambda(Y(t)) \underset{H_0}{\overset{H_1}{\gtrless}} \gamma \tag{21}$$

Using the likelihood ratio test, if $\Lambda(Y)$ exceeds γ, choose H_1, otherwise choose H_0 [6]. Figure 14 illustrates this concept:

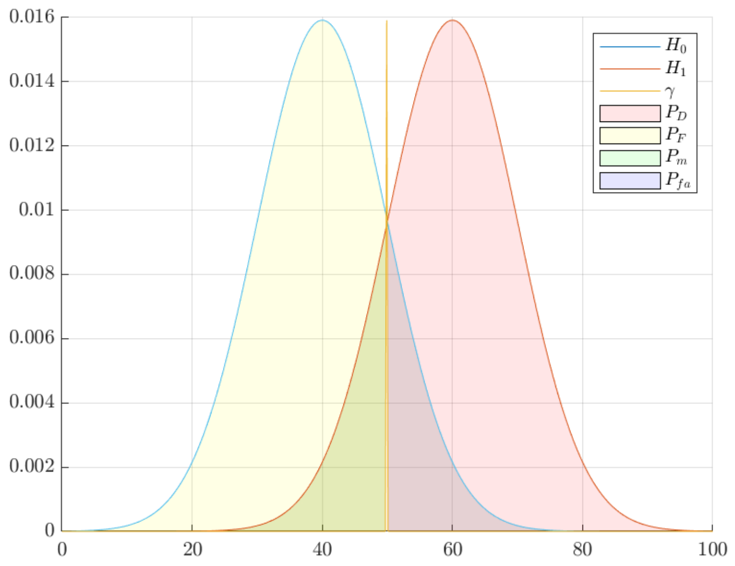


Figure 14: Illustration of the binary hypothesis. The left distribution is the probability that H_0 will occur. The right distribution is the probability that H_1 will occur. The vertical line in the middle is the decision threshold, γ.

2.4.2 Classical Detection Methods

There are many spectrum sensing detection methods available. They are broadly categorized as *cooperative*– where coordination between many receivers is leveraged to detect a waveform– or *non-cooperative*– where the receiver detects based only on what it can sense [5]. In this subsection, we will discuss three non-cooperative methods for detection.

2.4.2.1 Matched Filter Detector

If we assume the signal source $X(t)$ is deterministic and known to the receiver, then matched filtering is the optimal detection technique [5][6]. The matched filter, also known as a *coherent detector*, maximizes the signal-to-noise ratio (SNR) by correlating the received signal with the known waveform to determine the presence of the PU. While this technique performs well with AWGN, in the case of interference from a modulated signal, this technique performs poorly [5].

2.4.2.2 Energy Detector

If $X(t)$ and η are IID, then energy detection is the optimal detection technique. The energy detector treats H_0 and H_1 as varying levels of noise and determines the absence of a PU based on a defined threshold. This threshold is based on the noise power [5][6]. If there is no previous knowledge known about the signal source, this detector performs well. However, it performs poorly when classifying a signal's features, and it is vulnerable to unknown or changing noise levels.

2.4.2.3 Cyclostationary Feature Detector

Cyclostationary feature detectors (CFD) are able to differentiate between a modulated signal and noise, even at very low SNR values [5]. Let us reexamine Equation (18) and Equation (19) using Equation (17):

$$H_0 : S_X^v[k] = 0 \tag{22}$$

$$H_1 : S_X^v[k] \neq 0 \tag{23}$$

Now H_0 and H_1 can now be determined depending only on the CSD amplitude. If H_1 is nonzero at any k for a fixed v, a PU is active and exhibiting cyclostationary behavior of SR $R_s = v$. If H_0 is true, the PU is not active or not exhibiting cyclostationary behavior of SR $R_s = v$. Further, for each fixed v, the maximum k corresponds to the PUs CF $f_c = k$. In other words, the CFD allows us to distinguish between modulated signals and noise.

2.4.3 Test Statistic and Detection Threshold

There is no information known about the signal to be detected in a spectrum sensing environment, the probability it will be active, or if a signal even exists [6]. In this case, we cannot use our likelihood ratio developed in Equation (20); instead, we use what is known as a *sufficient statistic*. The sufficient statistic $l(S_X^v[k])$, or simply l, is a special case of the likelihood ratio that depends explicitly on $S_X^v[k]$. That is, knowing l is just as good as knowing the likelihood ratio [7]. For hypotheses Equations (22) and Equation (23) this results in the sufficient statistic for each (v, k) point of:

$$l = |S_X^v[k]|^2 \tag{24}$$

Notice that we still have not developed a threshold γ. Intuitively, γ should be proportional to the noise power σ_η^2. However, it is difficult to acquire noise power in a realistic spectrum sensing environment [6]. Many test statistics/thresholds have been studied for the CSD. Here, we will apply one of the simplest thresholds by generating the sufficient statistic of our received signal power and multiplying it by β.

$$\sigma_\eta^2 = \frac{1}{N}\sum_{n=0}^{N-1} |X[n]|^2 \tag{25}$$

The threshold is thus $\gamma = \beta\sigma_\eta^4$, where β is a scalar to adjust the sensitivity of false alarm [6]. The result of our threshold and the sufficient statistic leaves us with the test,

$$|S_X^v[k]|^2 \underset{H_0}{\overset{H_1}{\gtrless}} \beta\left(\frac{1}{N}\sum_{n=0}^{N-1} |X[n]|^2\right)^2 \tag{26}$$

Any cycle frequency with values above the threshold in Equation (26) is considered a detected waveform SR. Finally, a peak detection algorithm is employed for each SR to determine the CF of each waveform. This concept is illustrated in Figure 15.

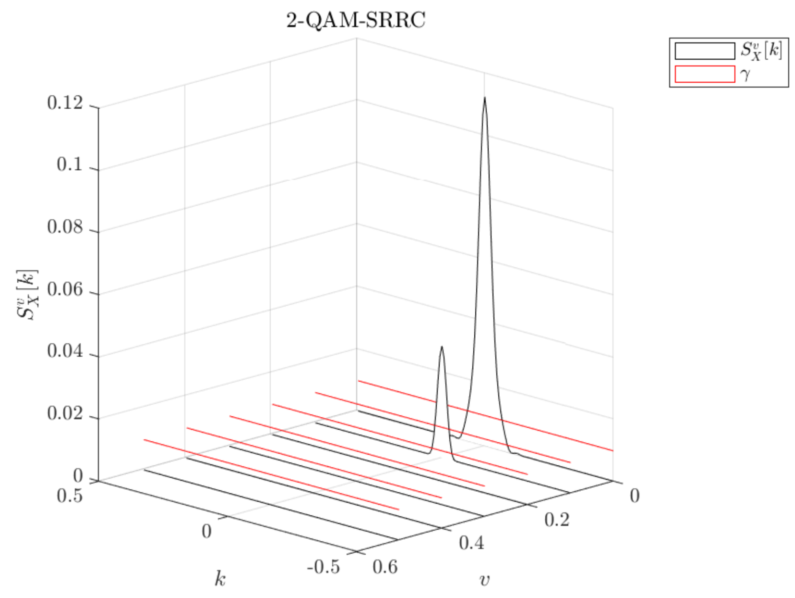


Figure 15: The application of Equation (26) to our waveform $x(t)$ from Figure 3. The sufficient statistic l produces the estimations for each candidate SR. To differentiate from noise, a threshold γ is applied using a β of 10^{-2}. PSD estimates with values above this threshold are determined to be detected SR slices. To determine the CF, the index of the peak of each detected SR slice is taken. The result is a detected waveform with indices (v, k), which map to the parameters (R_s, f_c).

CHAPTER III
IMPLEMENTATION

Our goal is to have the ability to perform spectrum sensing with an on-orbit CubeSat CR. To accomplish this, the CR must detect all waveform SRs and CFs for a received band. To differentiate these detected waveforms from noise, CFD is used. CFD can be realized using what is known as an *SCA*. Currently, SCAs are deployed for terrestrial applications. SCAs are computationally complex, so many FFTs are leveraged to increase performance. Each FFT needs BRAM for both the input and output equal to the number of frequencies N to be evaluated, resulting in high-BRAM use of $4N$. This high memory use is acceptable for terrestrial receivers, where memory is cheap and plentiful. In space, however, resources are minimal.

Since BRAM is overwhelmingly the limiting factor preventing the application of an SCA onto a CubeSat CR, if we can design an SCA with a low reliance on BRAM, we can perform spectrum sensing on-orbit. Time-domain processing reduces performance but allows for pipelined sample-by-sample processing techniques. To offset this speed reduction, instead of detecting all cyclic frequencies for all waveforms, we limit our application to only detect the SR and CF of M-QAM-SRRC waveforms. For this blind SR-CF estimator, we borrow the communications term *streaming*. Thus, this design will be referred to as the *streaming-SCA*.

3.1 Motivation

Traditional orbital architectures use either a single satellite or a constellation of satellites. Modern satellites increasingly use distributed orbital architectures such as fractionated, clustered, and trailing formations. These architectures operate using many coordinated small satellites instead of one large satellite. The launch of these small satellites, particularly CubeSats becoming increasingly common [8].

As space becomes more crowded with CubeSats that utilize crosslink and relay technology, the notion of inter-satellite interference is quickly realized as a concern. As with terrestrial applications, spectrum sensing is an efficient, low-cost technique to detect and avoid interference automatically.

As discussed in the previous chapter, SCAs are a fast, reliable, and effective method to perform spectrum sensing [1]. One significant weakness of SCAs, as noted in [1][6], is the requirement of large sample sizes, and therefore a large amount of memory to store those samples. The ability to quickly perform CFD is due to the high utilization of the memory-intensive FFT.

For example, if we evaluate the SSCA [1] for a complex-valued sample with a bit-width of $32b$ and 2^{16}-point FFTs, we find that we need at least $67Mb$ of memory per cycle frequency evaluated. Evaluating an nominal CubeSat CR FPGA's resources, the xc7z045fbg676-1, we find that it has $19.2Mb$ of BRAM– far below the SSCA's memory requirement of $67Mb$. Thus, a low-memory SCA is in need of design.

3.2 The Streaming-SCA

Let us assume that we wish to sense the waveforms present in a channel. As discussed in Chapter II, the CAF-DFT SCA is desirable because it does not rely on the FFT. In order to create a design that will fit onto a CubeSat CR, we will alter the CAF-DFT such that it:

- Evaluates only one pipelined (v, k) pair at a time, called a point. This is equivalent to one branch in Figure 12. If our SCA is well designed, branches are reconfigurable such that if extra resources are available on the platform, many instances can be used in parallel.

- Utilizes a minimum amount of BRAM. The CAF-DFT stores samples from the analog to digital converter (ADC) into BRAM. This is done to ensure every evaluation of a (v, k) pair is consistent when processed. For this design, we continuously acquire data directly from the ADC. Sample-by-sample acquisition risks inconsistency from one (v, k) to another, but as long as a signal is reasonably stable, this offers a massive savings of our CRs resources.
- Is designed for FPGA use. FPGAs offer greater SWaP efficiency, higher determinism and are more resilient in space environments.
- Does not evaluate the trivial case of $v = 0$, which is the spectral density of the conventional AF.
- Uses a fixed lowpass-filter (LPF) width of 2, corresponding to taps $[0.5, 0.5]$. The CAF-DFT utilizes a filter that dynamically changes with v. On an FPGA, this overhead is expensive, and therefore only 2 taps are used for the streaming-SCA.

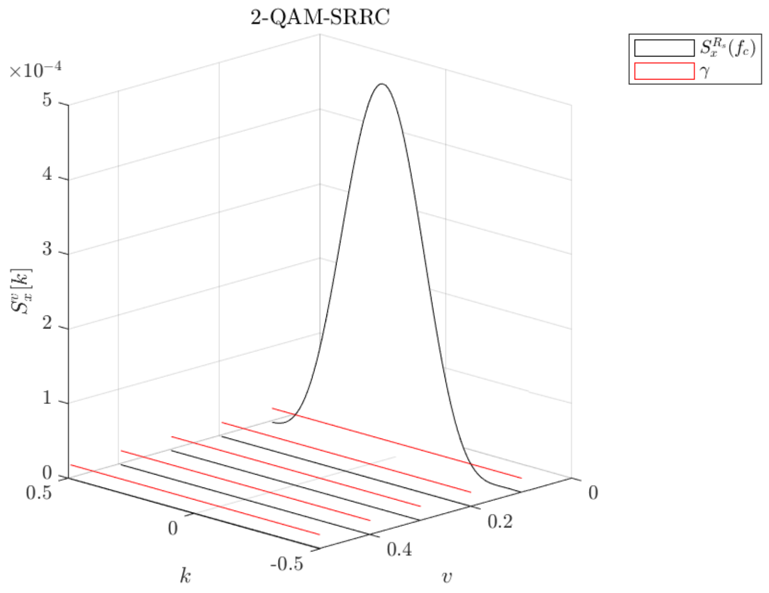


Figure 16: In this example we reproduce Figure 12, with the additional restrictions discussed above. We find that the peak of (v, k) perfectly matches the true SR and CF of $x(t)$ from Figure 3. Moreover, the peak exceeds the threshold $\gamma = \beta\sigma_\eta^4$, resulting in correct detection.

The streaming-SCA was modeled in MATLAB to produce Figure 16. We find that the streaming-SCA produces the same results as the TSM-SCA shown in Figure 4 with a scaling factor. The scaling results from the design choice to ensure the maximum possible value is never above one. This is necessary to prevent overflow in our digital design. MATLAB code for the streaming-SCA is provided in Appendix A.

In exchange for this low-memory utilization, the processing time is increased. This amounts to a design trade– parameters are determined depending on the SU's allotted time to move away from the PU. Delay is proportional to:

- SR step size– A larger step size increases processing speed, but also increases P_m.
- CF step size– A larger step size increases processing speed, but also increases

P_m.

- Sample size– A smaller sample size increases processing speed, but also increases P_m and P_{fa}.

3.3 Digital Design

To describe the implement of the streaming-SCA as a digital circuit, we will walk through each component and its subcomponents. This design uses no third-party intellectual property, which improves platform portability, reconfigurability, transparency of operation, and troubleshooting. There are minor references to VHDL functions, such as `resize()`, however, this chapter is intended to be language agnostic.

3.3.1 Center Frequency Estimator

The purpose of the center frequency estimator (CFE) is to produce a sufficient statistic distribution for each SR, CF candidate. This module produces the estimate of a (v, k) pair, which maps to some (R_s, f_c). Mathematically, this module corresponds to Equation (17), rewritten below, for some $X, v, k, n,$.

$$S_X^v[k] = \frac{1}{N}\sum_{n=0}^{N-1} \left|(X[n]e^{-j2\pi kn}) * h[n]\right|^2 e^{-j2\pi vn}$$

The mixer is controlled by FCW_k, corresponding to the heterodyned expression $X[n]e^{-j2\pi kn}$. Next, a LPF module smooths the mixed signal with the fixed taps $[0.5, 0.5]$, corresponding to the channelized expression $(X[n]e^{-j2\pi kn}) * h[n]$. After this, the CAF is applied, corresponding to the CFE equation
$S_X^v[k] = \frac{1}{N}\sum_{n=0}^{N-1} \left|(X[n]e^{-j2\pi kn}) * h[n]\right|^2 e^{-j2\pi vn}$. Lastly, the magnitude squared is calculated, resulting in one poiny of the sufficient statistic summation $l = |S_x^v[k]|^2$ for one set of X, v, k, n values.

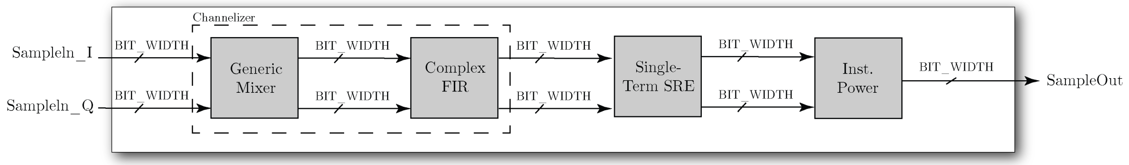


Figure 17: Digital design of the CFE module.

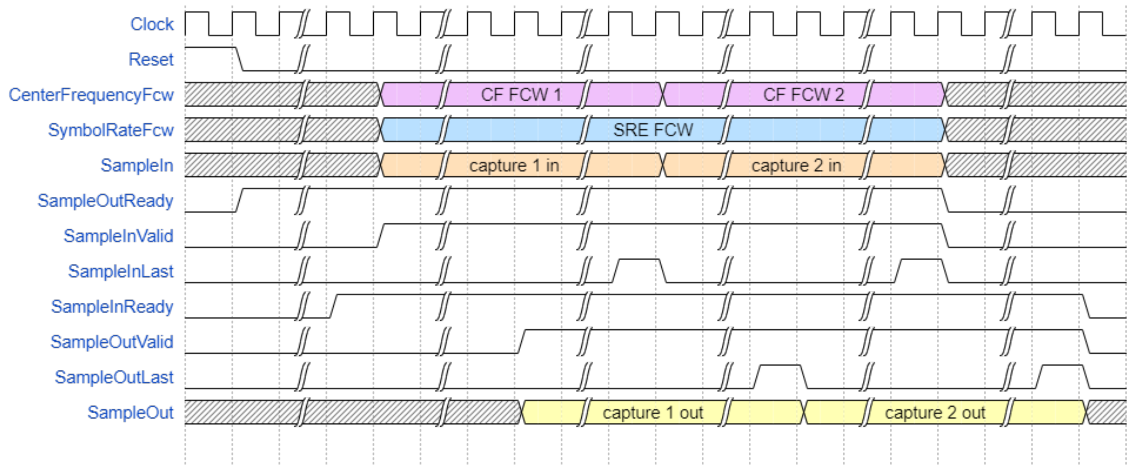


Figure 18: Timing diagram of the CFE module.

Table 1: Pinout of the CFE module.

Port Name	**Direction**	**Data Type**	**Description**
N_SAMPLES	Generic	natural	Number of samples accumulated.
BIT_WIDTH	Generic	natural	Format of samples is s0.(BIT_WIDTH-1)
Clock	Input	std_logic	Synchronous clock.
Reset	Input	std_logic	Asynchronous reset.
SampleOutReady	Input	std_logic	Asserted when downstream module is ready to accept input.
SampleInValid	Input	std_logic	High when input is valid.
SampleInLast	Input	std_logic	Delineates between CF, SR pairs being processed.
CenterFrequencyFcw	Input	signed(16)	CF to estimate.
SymbolRateFcw	Input	signed(16)	SR to to estimate.
SampleIn_I	Input	signed(BIT_WIDTH)	Real component of sample.
SampleIn_Q	Input	signed(BIT_WIDTH)	Imaginary component of sample.
SampleInReady	Output	std_logic	Asserted when module is ready to accept input.
SampleOutValid	Output	std_logic	High when output is valid.
SampleOutLast	Output	std_logic	Delineates between CF, SR pairs being processed.
OverflowStatus	Output	std_logic_vector(4)	[3]: CF mixer I [2]: CF mixer Q [1]: SR mixer I [0]: SR mixer Q
SampleOut	Output	signed(BIT_WIDTH)	Real-valued result.

3.3.2 Symbol Rate Estimator

The purpose of the SR estimator (SRE) is to estimate the correlation between a waveform and the SR cycle frequency v. Mathematically, this module corresponds to the CAF equation evaluated at $\tau = 0$.

$$R_X^v[0] = \frac{1}{N}\sum_{n=0}^{N-1} |X[n]|^2 e^{-j2\pi vn} dt$$

The instantaneous power module corresponds to the magnitude squared $|X[n]|^2$. After, the DFT is applied, corresponding to the CAF $R_X^v[0] = \frac{1}{N}\sum_{n=0}^{N-1} |X[n]|^2 e^{-j2\pi vn} dt$.

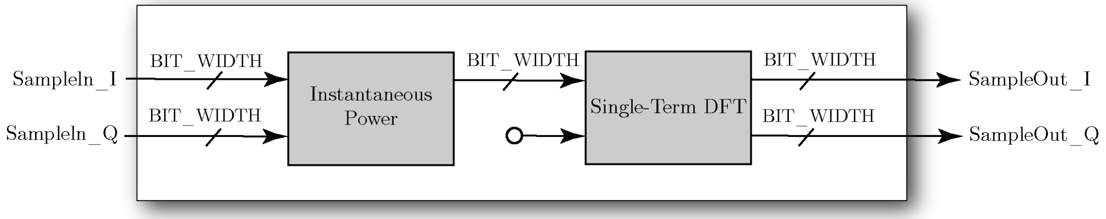


Figure 19: Digital design of the SRE module.

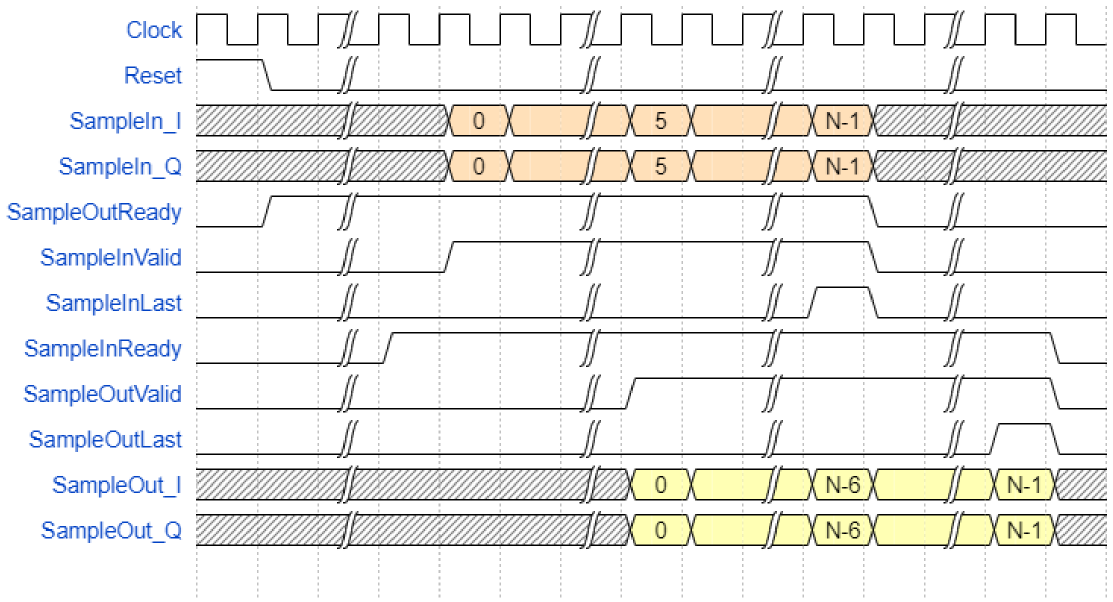


Figure 20: Timing diagram of the SRE module.

Table 2: Pinout of the SRE module.

Port Name	**Direction**	**Data Type**	**Description**
N_SAMPLES	Generic	natural	Number of samples accumulated.
BIT_WIDTH	Generic	natural	Format of samples is s0.(BIT_WIDTH-1)
Clock	Input	std_logic	Synchronous clock.
Reset	Input	std_logic	Asynchronous reset.
SampleOutReady	Input	std_logic	Asserted when downstream module is ready to accept input.
SampleInValid	Input	std_logic	High when input is valid.
SampleInLast	Input	std_logic	Delineates between SRs being processed.
OscillatorFcw	Input	signed(16)	SR to to estimate.
SampleIn_I	Input	signed(BIT_WIDTH)	Real component of sample.
SampleIn_Q	Input	signed(BIT_WIDTH)	Imaginary component of sample.
SampleInReady	Output	std_logic	Asserted when module is ready to accept in-put.
SampleOutValid	Output	std_logic	High when output is valid.
SampleOutLast	Output	std_logic	Delineates between SRs being processed.
OverflowStatus	Output	std_logic_vector(2)	[1]: mixer I [0]: mixer Q
SampleOut_I	Output	signed(BIT_WIDTH)	Real component of sample.
SampleOut_Q	Output	signed(BIT_WIDTH)	Imaginary component of sample.

3.3.3 Discrete Fourier Transform

The DFT is used to perform frequency-domain analysis of discrete-time signals. For this implementation, the below expression is evaluated for one particular k at a time, as opposed to the FFT, which calculates all values of k simultaneously. The DFT of $X[n]$ is expressed mathematically as:

$$\hat{X}[k] = \sum_{n=0}^{N-1} X[n]e^{-j2\pi vn}$$

Here, $\hat{X}[k]$ is the series of Fourier coefficients of the signal $X[n]$. The mixer is controlled by FCW_k, corresponding to the heterodyned expression $X[n]e^{-j2\pi kn}$. The heterodynes are then accumulated for N samples, corresponding to the DFT

$\hat{X}[k] = \sum_{n=0}^{N-1} X[n]e^{-j2\pi kn}$.

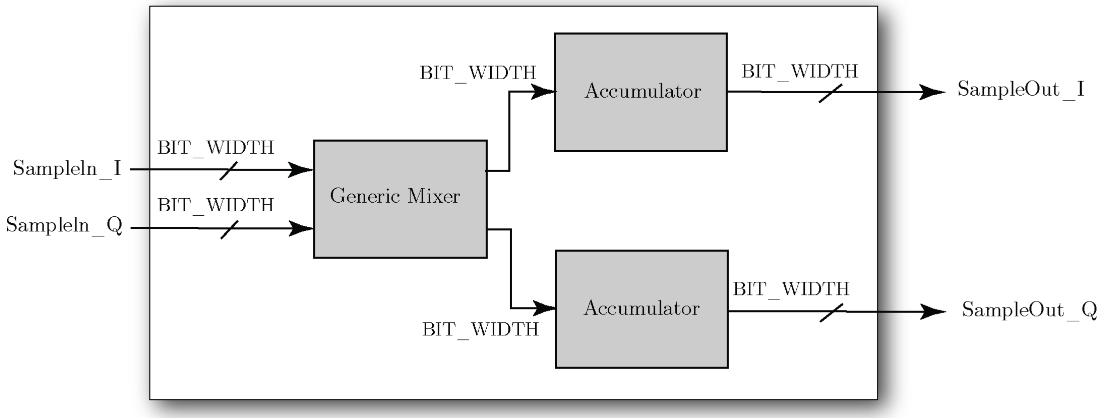


Figure 21: Discrete Fourier transform block diagram module.

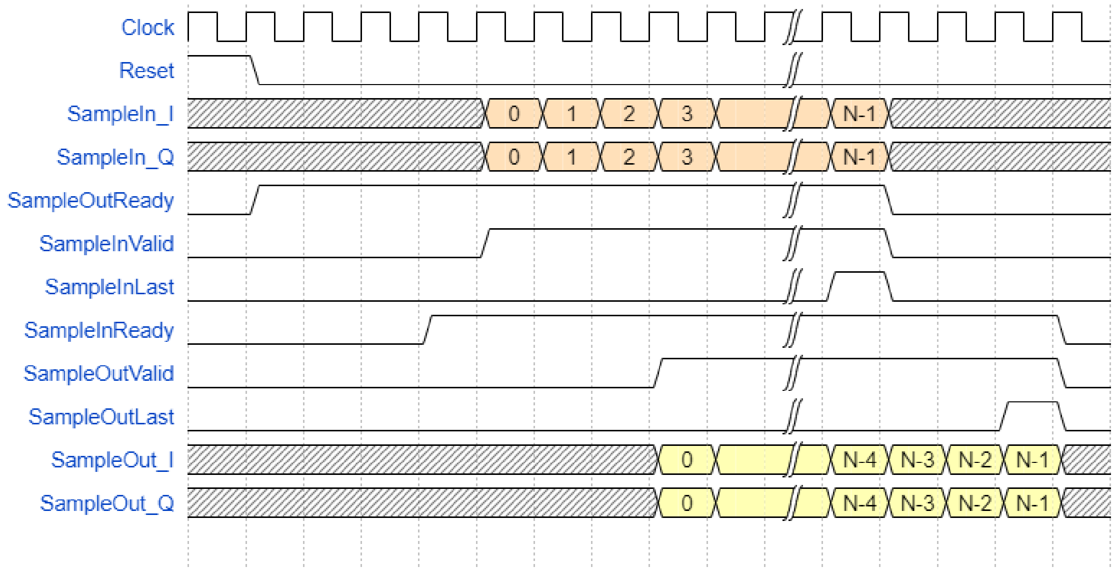


Figure 22: Timing diagram of the DFT module.

Table 3: Pinout of the DFT module.

Port Name	**Direction**	**Data Type**	**Description**
N_SAMPLES	Generic	natural	Number of samples accumulated.
BIT_WIDTH	Generic	natural	Format of samples is s0.(BIT_WIDTH-1)
Clock	Input	std_logic	Synchronous clock.
Reset	Input	std_logic	Asynchronous reset.
SampleOutReady	Input	std_logic	Asserted when downstream module is ready to accept input.
SampleInValid	Input	std_logic	High when input is valid.
SampleInLast	Input	std_logic	Delineates between frequencies being processed.
OscillatorFcw	Input	signed(16)	Frequency to estimate.
SampleIn_I	Input	signed(BIT_WIDTH)	Real component of sample.
SampleIn_Q	Input	signed(BIT_WIDTH)	Imaginary component of sample.
SampleInReady	Output	std_logic	Asserted when module is ready to accept input.
SampleOutValid	Output	std_logic	High when output is valid.
SampleOutLast	Output	std_logic	Delineates between frequencies being processed.
OverflowStatus	Output	std_logic_vector(2)	[1]: mixer I [0]: mixer Q
SampleOut_I	Output	signed(BIT_WIDTH)	Real component of sample.
SampleOut_Q	Output	signed(BIT_WIDTH)	Imaginary component of sample.

3.3.4 Frequency Mixer

A *heterodyne* is a signal frequency that is produced by mixing the frequencies of two signals. In this context, frequency mixing is performed by element-wise multiplication of an signal with a digital numerically controlled local oscillator (NCO). The frequency mixer is mathematically expressed as:

$$e^{-j2\pi(k_1+k_2)n} = e^{-j2\pi k_1 n} \cdot e^{-j2\pi k_2 n}$$

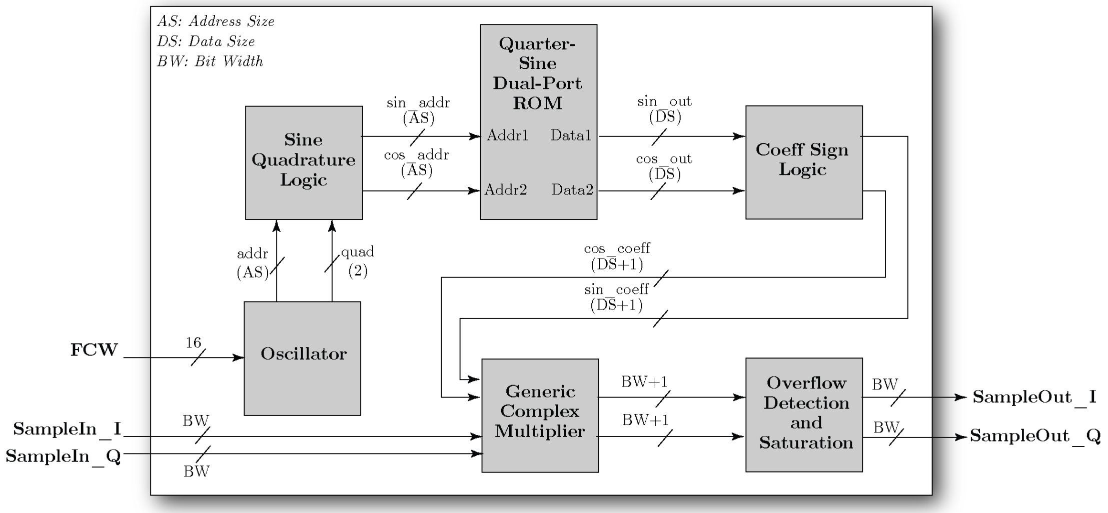


Figure 23: Digital design of the frequency mixer module.

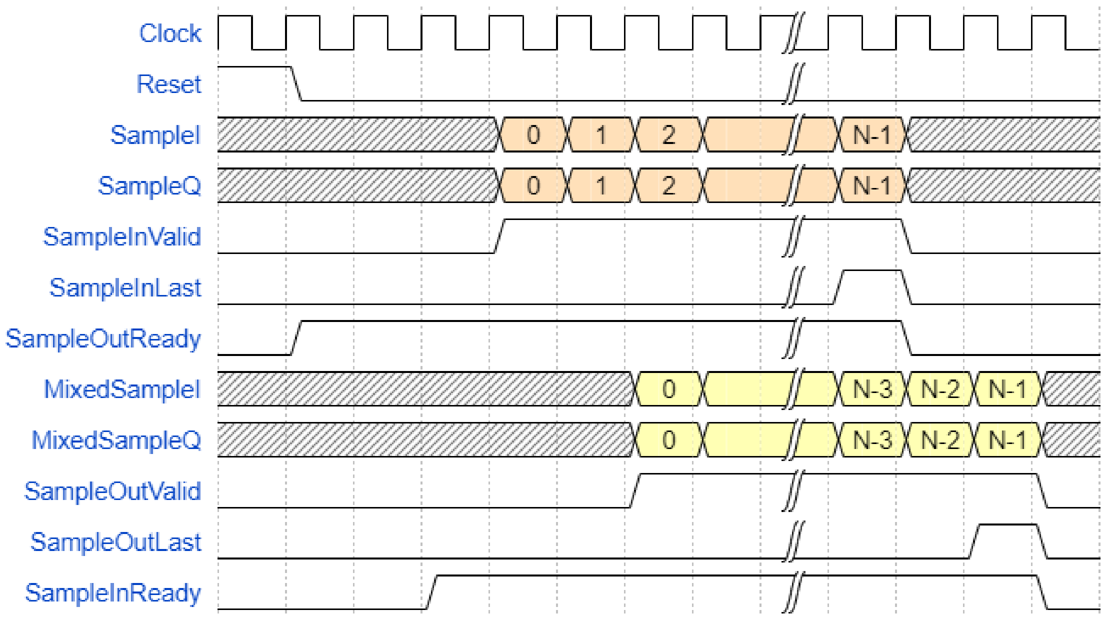


Figure 24: Timing diagram of the frequency mixer module.

Table 4: Pinout of the frequency mixer module.

Port Name	**Direction**	**Data Type**	**Description**
BIT_WIDTH	Generic	natural	Format of samples is s0.(BIT_WIDTH-1)
Clock	Input	std_logic	Synchronous clock.
Reset	Input	std_logic	Asynchronous reset.
SampleOutReady	Input	std_logic	Asserted when downstream module is ready to accept input.
SampleInValid	Input	std_logic	High when input is valid.
SampleInLast	Input	std_logic	Pass through.
OscillatorFcw	Input	signed(16)	Frequency to estimate.
SampleIn_I	Input	signed(BIT_WIDTH)	Real component of sample.
SampleIn_Q	Input	signed(BIT_WIDTH)	Imaginary component of sample.
SampleInReady	Output	std_logic	Asserted when module is ready to accept input.
SampleOutValid	Output	std_logic	High when output is valid.
SampleOutLast	Output	std_logic	Pass through.
OverflowStatus	Output	std_logic_vector(2)	[1]: I [0]: Q
SampleOut_I	Output	signed(BIT_WIDTH)	Real component of sample.
SampleOut_Q	Output	signed(BIT_WIDTH)	Imaginary component of sample.

3.3.5 Low-Pass Filter

A finite impulse response (FIR) is a filter whose response to any finite-duration input settles to zero in a finite amount of time. Filter coefficients are calculated given the desired properties of the filter in MATLAB or python. The FIR implements the convolution sum as defined by the equation:

$$Y[n] = \sum_{i=0}^{N-1} b[i]X[n-i]$$

Here, $b[k]$ are the filter coefficients for an N^{th}-order filter with $0 \leq i < N$ and $X[n]$ is the series of input samples. For this design a fixed tap length of 2 is used, which corresponds to the low-pass filter coefficients $b[0] = 0.5$, $b[1] = 0.5$.

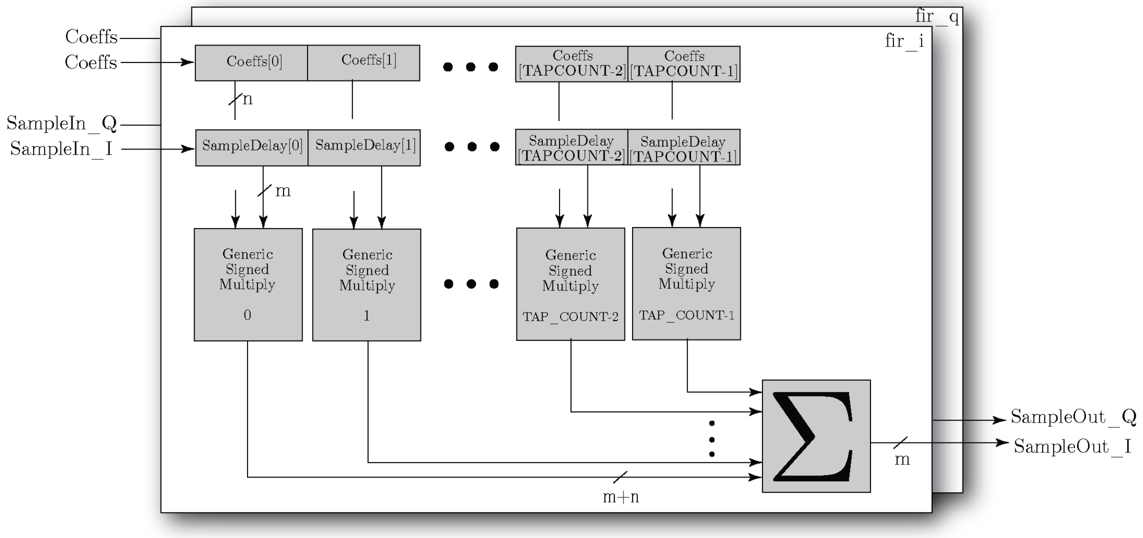


Figure 25: Digital design of the LPF module.

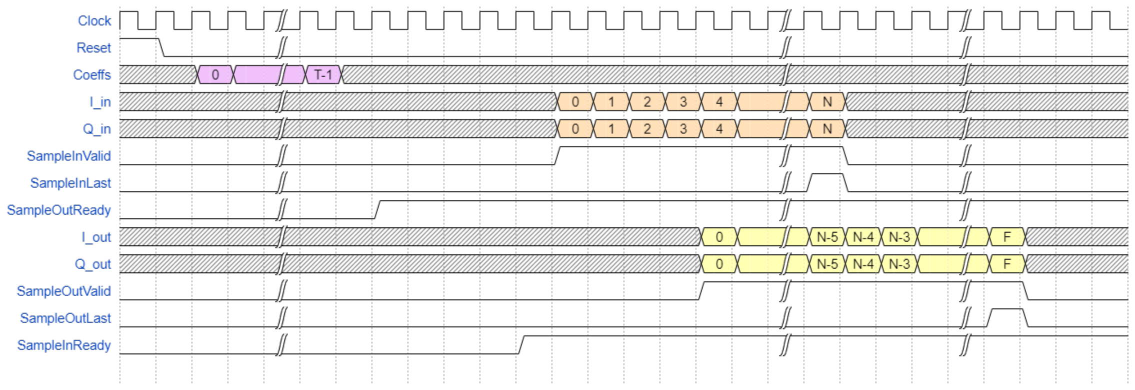


Figure 26: Timing diagram of the low-pass filter module.

Table 5: Pinout of the low-pass filter module.

Port Name	**Direction**	**Data Type**	**Description**
TAP_COUNT	Generic	natural	Number of Coeffs.
TAP_WIDTH	Generic	natural	Format of coeffecients is s0.(TAP_WIDTH-1)
BIT_WIDTH	Generic	natural	Format of samples is s0.(BIT_WIDTH-1)
Clock	Input	std_logic	Synchronous clock.
Reset	Input	std_logic	Asynchronous reset.
Coeffs	Input	signed array(TAP_LENGTH)	Fixed-values of $b[0] = 0.5$, $b[1] = 0.5$ for this application.
SampleOutReady	Input	std_logic	Asserted when downstream module is ready to accept input.
SampleInValid	Input	std_logic	High when input is valid.
SampleInLast	Input	std_logic	Pass through.
SampleIn_I	Input	signed(BIT_WIDTH)	Real component of sample.
SampleIn_Q	Input	signed(BIT_WIDTH)	Imaginary component of sample.
SampleInReady	Output	std_logic	Asserted when module is ready to accept input.
SampleOutValid	Output	std_logic	High when output is valid.
SampleOutLast	Output	std_logic	Pass through.
SampleOut_I	Output	signed(BIT_WIDTH)	Real component of sample.
SampleOut_Q	Output	signed(BIT_WIDTH)	Imaginary component of sample.

3.3.6 Instantaneous Power

This module serves as a special case of the AF for $\tau = 0$. This special case is equivalent to one index of Parseval's theorem $|X[n]|^2$, which is expressed as the magnitude squared. This is mathematically expressed as:

$$\begin{aligned} P_X[n] &= |X[n]|^2 \\ &= ((\Re\{X[n]\}^2 + \Im\{X[n]\}^2)^{\frac{1}{2}})^2 \\ &= \Re\{X[n]\}^2 + \Im\{X[n]\}^2 \end{aligned}$$

Here, the real and imaginary components are squared in parallel and then added.

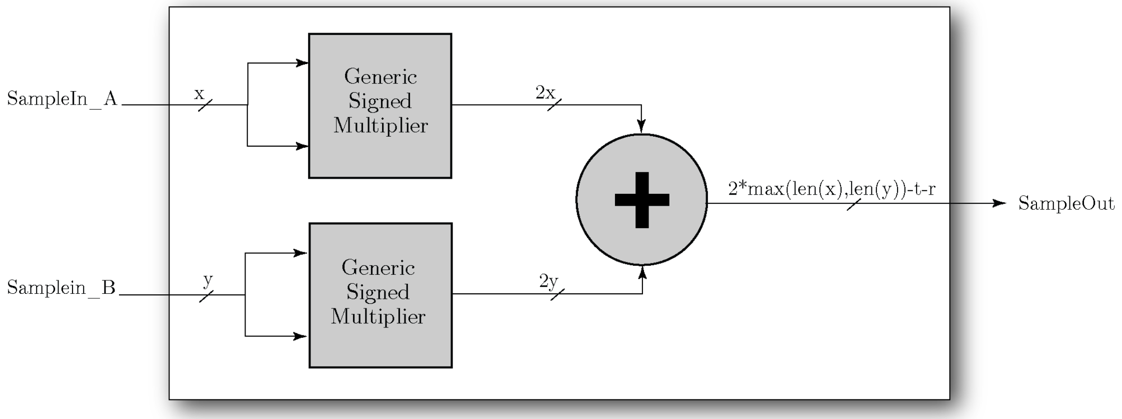


Figure 27: Digital design of the instantaneous power module.

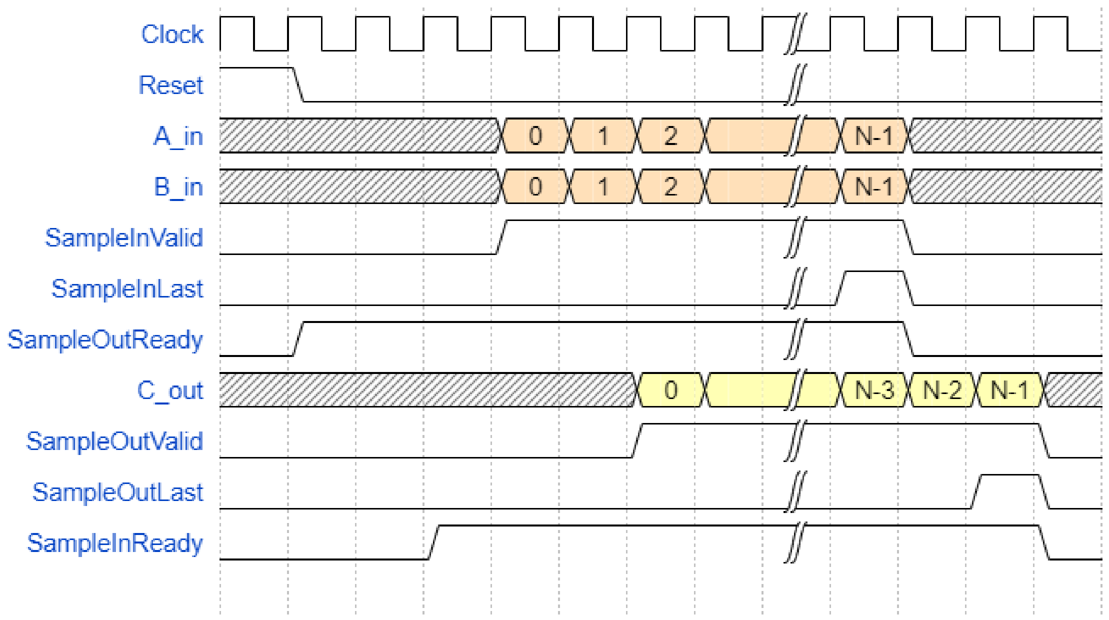


Figure 28: Timing diagram of the instantaneous power module.

Table 6: Pinout of the instantaneous power module.

Port Name	**Direction**	**Data Type**	**Description**
BIT_WIDTH	Generic	natural	Format of samples is s0.(BIT_WIDTH-1)
LATENCY	Generic	natural	Control total latency of operation.
TRUNC	Generic	natural	Remove extraneous MSBs.
RND	Generic	natural	Round LSB.
Clock	Input	std_logic	Synchronous clock.
Reset	Input	std_logic	Asynchronous reset.
SampleOutReady	Input	std_logic	Asserted when downstream module is ready to accept input.
SampleInValid	Input	std_logic	High when input is valid.
SampleInLast	Input	std_logic	Pass through.
SampleIn_I	Input	signed(BIT_WIDTH)	Real component of sample.
SampleIn_Q	Input	signed(BIT_WIDTH)	Imaginary component of sample.
SampleInReady	Output	std_logic	Asserted when module is ready to accept input.
SampleOutValid	Output	std_logic	High when output is valid.
SampleOutLast	Output	std_logic	Pass through.
SampleOut	Output	signed(2*BIT_WIDTH -RND-TRUNC)	Real-valued output.

3.3.7 Complex Multiplier

The product of two complex numbers can be expressed in terms of pairs of real and imaginary components as follows:

$$X_1[n]X_2[n] = (A + Bj)(C + Dj)$$
$$\Re\{X_1[n]X_2[n]\} = AC - BD$$
$$\Im\{X_1[n]X_2[n]\} = AD + BC$$

Here, the multiplication of complex values $X_1[n]X_2[n]$ simplifies to four real multiplications, one real addition, and one real subtraction.

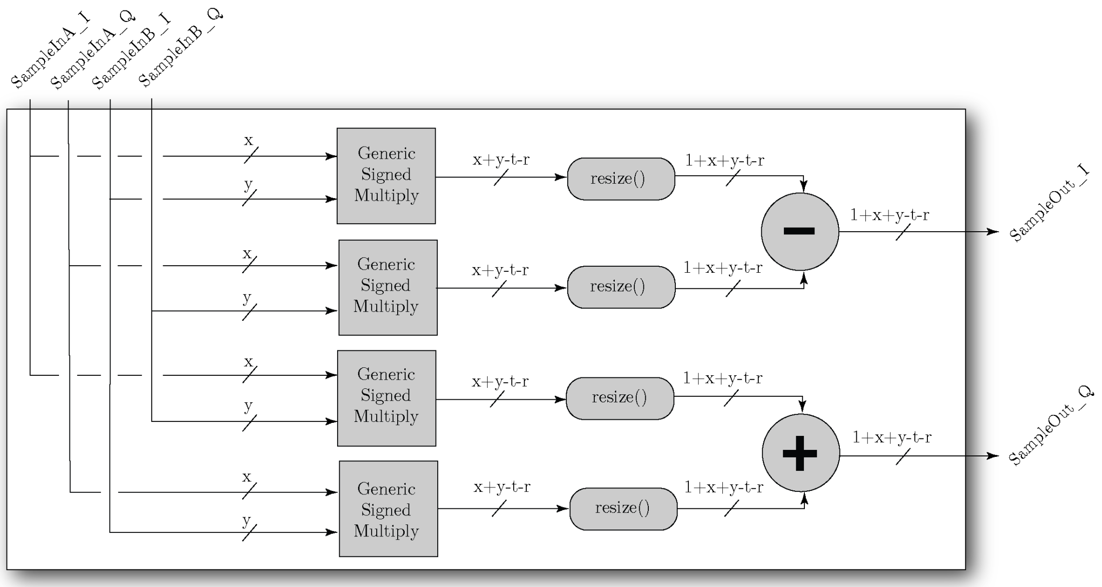


Figure 29: Digital design of the complex multiplier module.

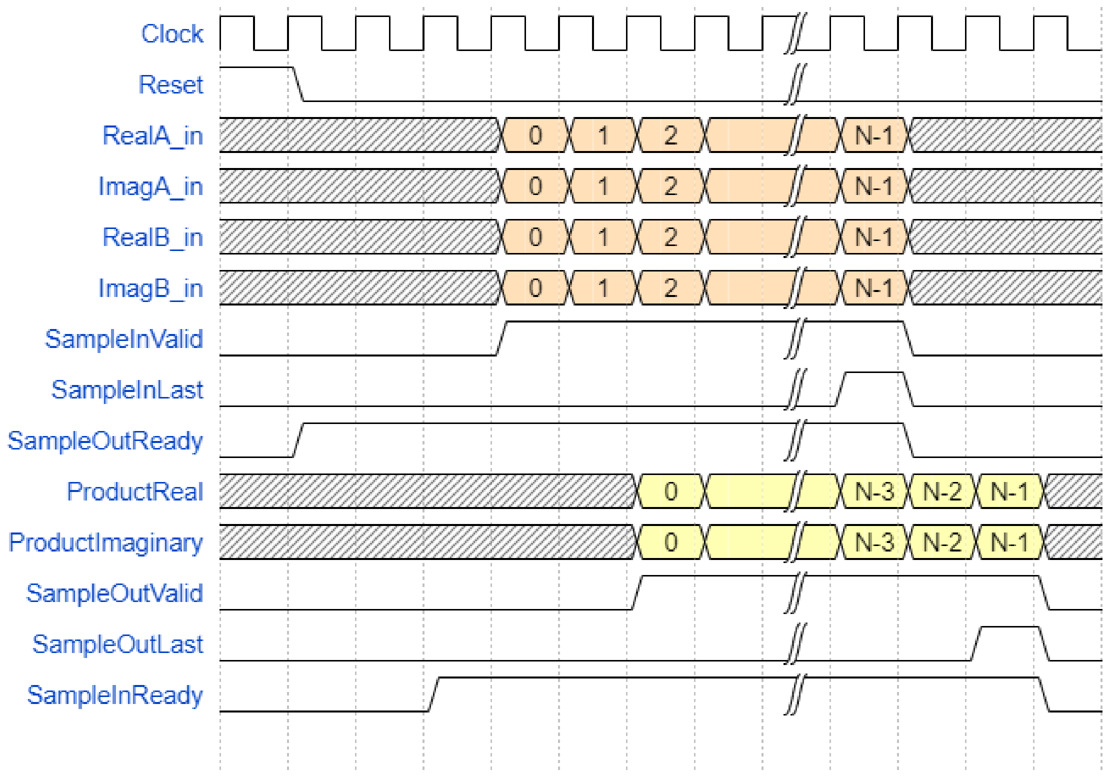


Figure 30: Timing diagram of the complex multiplier module.

Table 7: Pinout of complex multiplier module.

Port Name	**Direction**	**Data Type**	**Description**
LEN_A	Generic	natural	Format of samples is s0.(BIT_WIDTH-1)
LEN_B	Generic	natural	Format of samples is s0.(BIT_WIDTH-1)
LATENCY	Generic	natural	Control total latency of operation.
TRUNC	Generic	natural	Remove extraneous MSBs.
RND	Generic	natural	Round LSB.
Clock	Input	std_logic	Synchronous clock.
Reset	Input	std_logic	Asynchronous reset.
SampleOutReady	Input	std_logic	Asserted when downstream module is ready to accept input.
SampleInValid	Input	std_logic	High when input is valid.
SampleInLast	Input	std_logic	Pass through.
SampleIn_I_A	Input	signed(LEN_A)	Real component of sample A.
SampleIn_Q_A	Input	signed(LEN_A)	Imaginary component of sample A.
SampleIn_I_B	Input	signed(LEN_B)	Real component of sample B.
SampleIn_Q_B	Input	signed(LEN_B)	Imaginary component of sample B.
SampleInReady	Output	std_logic	Asserted when module is ready to accept input.
SampleOutValid	Output	std_logic	High when output is valid.
SampleOutLast	Output	std_logic	Pass through.
SampleOut_I	Output	signed(LEN_A +LEN_B -RND-TRUNC+1)	Real component of sample.
SampleOut_Q	Output	signed(LEN_A +LEN_B -RND-TRUNC+1))	Imaginary component of sample.

3.3.8 Multiplier

There are many real multipliers used throughout this design. Although multipliers exist as a primitive operation in HDL languages, a module is used to ensure proper timing, handshaking, bit-widths, delays, fixed-point control, and rounding. Mathematically, this module performs the multiplication of two real values, expressed as:

$$Y[n] = X_1[n]X_2[n] \tag{27}$$

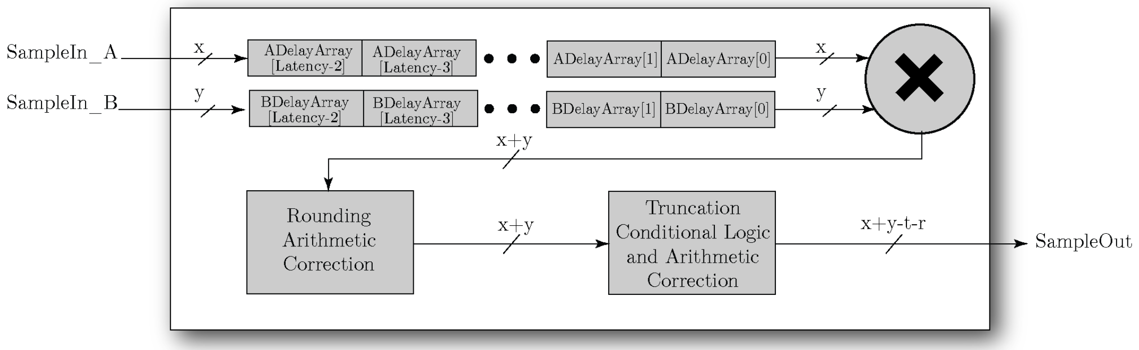


Figure 31: Digital design of the multiplier module.

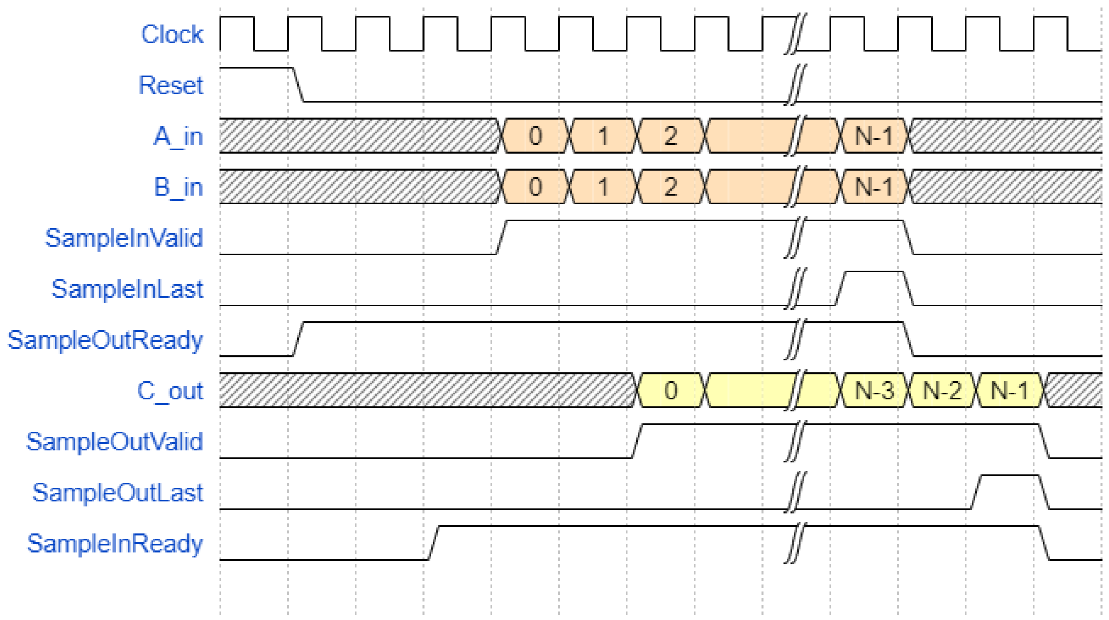


Figure 32: Timing diagram of the multiplier module.

Table 8: Pinout of multiplier module.

Port Name	**Direction**	**Data Type**	**Description**
LEN_A	Generic	natural	Format of samples is s0.(BIT_WIDTH-1)
LEN_B	Generic	natural	Format of samples is s0.(BIT_WIDTH-1)
LATENCY	Generic	natural	Control total latency of operation.
TRUNC	Generic	natural	Remove extraneous MSBs.
RND	Generic	natural	Round LSB.
Clock	Input	std_logic	Synchronous clock.
Reset	Input	std_logic	Asynchronous reset.
SampleOutReady	Input	std_logic	Asserted when downstream module is ready to accept input.
SampleInValid	Input	std_logic	High when input is valid.
SampleInLast	Input	std_logic	Pass through.
SampleIn_A	Input	signed(LEN_A)	Real-valued sample A.
SampleIn_B	Input	signed(LEN_B)	Real-valuedsample B.
SampleInReady	Output	std_logic	Asserted when module is ready to accept input.
SampleOutValid	Output	std_logic	High when output is valid.
SampleOutLast	Output	std_logic	Pass through.
SampleOut	Output	signed(LEN_A +LEN_B-RND -TRUNC)	Real-valued sample out.

3.3.9 Accumulator

The accumulator is the most important module in the design concerning flow control. This module has three core functions: Acting as the summation component of all previous modules, acting as the normalization component of all previous components, and acting as the reset between sample sets. Mathematically, the accumulator is expressed as the difference equation:

$$Y[n] = \begin{cases} Y[n] = X[n], & if\ n = 0 \\ Y[n] = Y[n-1] + X[n], & otherwise \end{cases} \tag{28}$$

Where $X[n]$ and $Y[n]$ are the input and output, respectively, at the discrete point in time n, and $Y[n-1]$ is the sum of all previous discrete samples in $X[n]$. This implementation accumulates discrete samples until a control signal is received, instructing the accumulator to reset the running sum. Notably, this is performed with a zero-clock delay between the last sample of one sequence and the first sample of the following sequence.

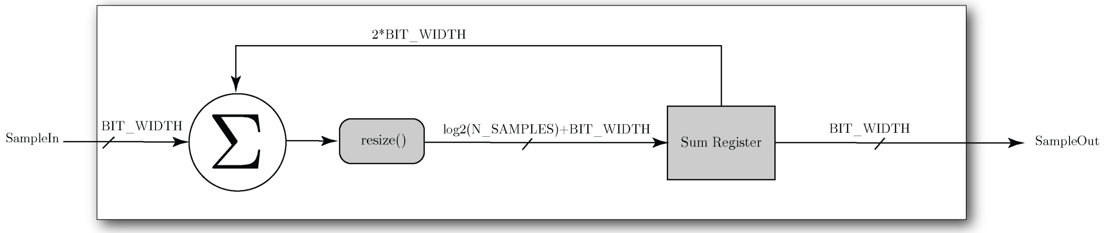


Figure 33: Digital design of the accumulator module.

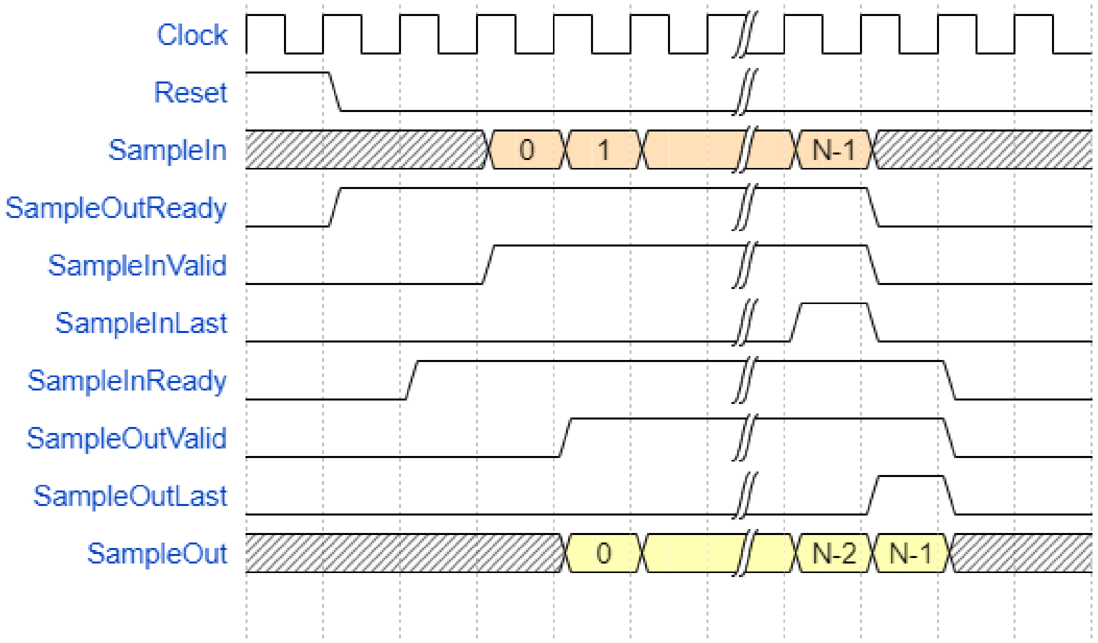


Figure 34: Timing diagram of the accumulator module.

Table 9: Pinout of the accumulator module.

Port Name	**Direction**	**Data Type**	**Description**
N_SAMPLES	Generic	natural	Number of samples accumulated.
BIT_WIDTH	Generic	natural	Format of samples is s0.(BIT_WIDTH-1)
Clock	Input	std_logic	Synchronous clock.
Reset	Input	std_logic	Asynchronous reset.
SampleOutReady	Input	std_logic	Asserted when downstream module is ready to accept input.
SampleInValid	Input	std_logic	High when input is valid.
SampleInLast	Input	std_logic	Delineates between CF, SR pairs being processed.
SampleIn_I	Input	signed(BIT_WIDTH)	Real component of sample.
SampleIn_Q	Input	signed(BIT_WIDTH)	Imaginary component of sample.
SampleInReady	Output	std_logic	Asserted when module is ready to accept input.
SampleOutValid	Output	std_logic	High when output is valid.
SampleOutLast	Output	std_logic	Delineates between CF, SR pairs being processed.
SampleOut	Output	signed(BIT_WIDTH)	Real-valued result.

CHAPTER IV
PERFORMANCE CHARACTERIZATION

This chapter will characterize the performance of the streaming SCA's ability to correctly sense and classify the CF and SR of signals in a monitored spectrum. First, we generate a transmit M-QAM-SRRC waveform in software using python. The waveform is then passed through a software channel and saved to a file in a fixed-point format using python. The receiver is a VHDL testbench that reads the file, processes the in-phase and quadrature (IQ) samples using the streaming SCA, then saves CSD results to a new file. We then plot the CSD in MATLAB, along with its corresponding threshold. A detected SR is any spectral slice v above the threshold. A detected CF is a peak k for a detected SR slice. Finally, these bins are mapped to their true values (R_s, f_c), respectively.

4.1 Experiments

Characterization is performed by first establishing baseline parameters for the transmitter and receiver. While baseline parameters provide results that are easy to be accurately detected, they may not reflect real-world constraints such as hardware restrictions or channel impairments. Each subsequent experiment is a parametric analysis of the effect of modifying a given parameter. The parameters evaluated are shown in Table 10 and Table 11.

Table 10: Python transmitter & channel
Here, $f_s = 1MHz$ for all experiments, and r is rolloff.

Expt.	$\boldsymbol{M}$	$\boldsymbol{r}$	$\boldsymbol{R_s\,(MBd)}$	$\boldsymbol{f_c\,(MHz)}$	$\boldsymbol{\frac{E_b}{N_0}\,(dB)}$
I	4	0.35	0.10	0.05	14.00
II	16	0.35	0.10	0.05	14.00
III	4	0.20	0.10	0.05	14.00
IV	4	0.35	0.01	0.05	14.00
V	4	0.35	0.10	-0.05	14.00
VI	4	0.35	0.10	0.05	9.00
VII	4	0.35	0.10	0.05	14.00
VIII	4	0.35	0.10	0.05	14.00
IX	4	0.35	0.10	0.05	14.00
X	4	0.35	0.10	0.05	14.00
XI	4	0.35	0.10	0.05	14.00
XII	4; 16	0.35; 0.35	0.10; 0.25	-0.20; 0.20	14.00
XIII	4; 16	0.35; 0.35	0.10; 0.25	0.05; 0.10	14.00

Table 11: VHDL receiver
Here, BW is bit-width, (v, k) are the search space.

Expt.	$\boldsymbol{f_s\,(MHz)}$	$\boldsymbol{N}$	$\boldsymbol{BW}$	$\boldsymbol{v}$	$\boldsymbol{k}$
I	1.00	2^{16}	16	$[R_s : R_s : \frac{F_s}{2}]$	$[-\frac{F_s}{2} : f_c : \frac{F_s}{2}]$
II	1.00	2^{16}	16	$[R_s : R_s : \frac{F_s}{2}]$	$[-\frac{F_s}{2} : f_c : \frac{F_s}{2}]$
III	1.00	2^{16}	16	$[R_s : R_s : \frac{F_s}{2}]$	$[-\frac{F_s}{2} : f_c : \frac{F_s}{2}]$
IV	1.00	2^{16}	16	$[R_s : R_s : \frac{F_s}{2}]$	$[-\frac{F_s}{2} : f_c : \frac{F_s}{2}]$
V	1.00	2^{16}	16	$[R_s : R_s : \frac{F_s}{2}]$	$[-\frac{F_s}{2} : f_c : \frac{F_s}{2}]$
VI	1.00	2^{16}	16	$[R_s : R_s : \frac{F_s}{2}]$	$[-\frac{F_s}{2} : f_c : \frac{F_s}{2}]$
VII	10.00	2^{16}	16	$[1MHz : 1MHz : \frac{F_s}{2}]$	$[-\frac{F_s}{2} : 50kHz : \frac{F_s}{2}]$
VIII	1.00	2^{12}	16	$[R_s : R_s : \frac{F_s}{2}]$	$[-\frac{F_s}{2} : f_c : \frac{F_s}{2}]$
IX	1.00	2^{16}	8	$[R_s : R_s : \frac{F_s}{2}]$	$[-\frac{F_s}{2} : f_c : \frac{F_s}{2}]$
X	1.00	2^{16}	16	R_s	$[-\frac{F_s}{2} : 10kHz : \frac{F_s}{2}]$
XI	1.00	2^{16}	16	$[99.9kHz : 1Hz : 100.1kHz]$	f_c
XII	1.00	2^{16}	16	$[50kHz : 50kHz : \frac{F_s}{2}]$	$[-\frac{F_s}{2} : 50kHz : \frac{F_s}{2}]$
XIII	1.00	2^{16}	16	$[50kHz : 50kHz : \frac{F_s}{2}]$	$[-\frac{F_s}{2} : 50kHz : \frac{F_s}{2}]$

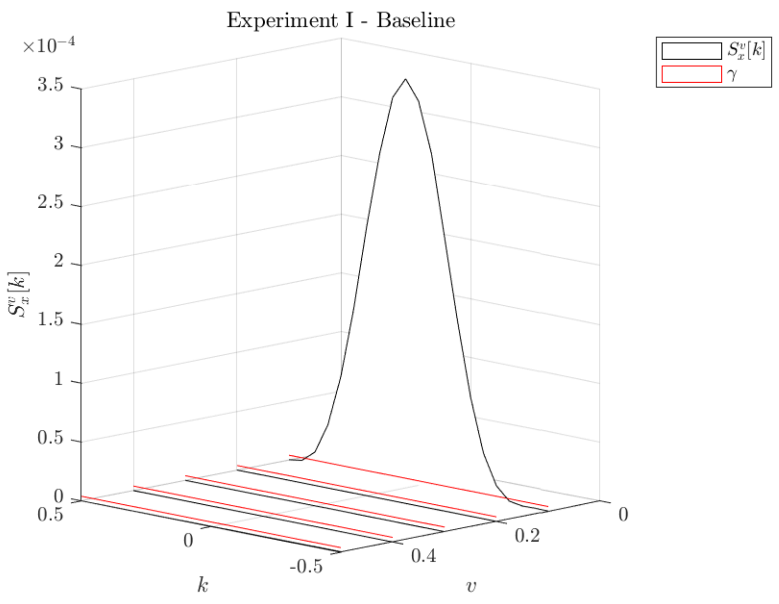


Figure 35: CSD estimate plot for experiment I

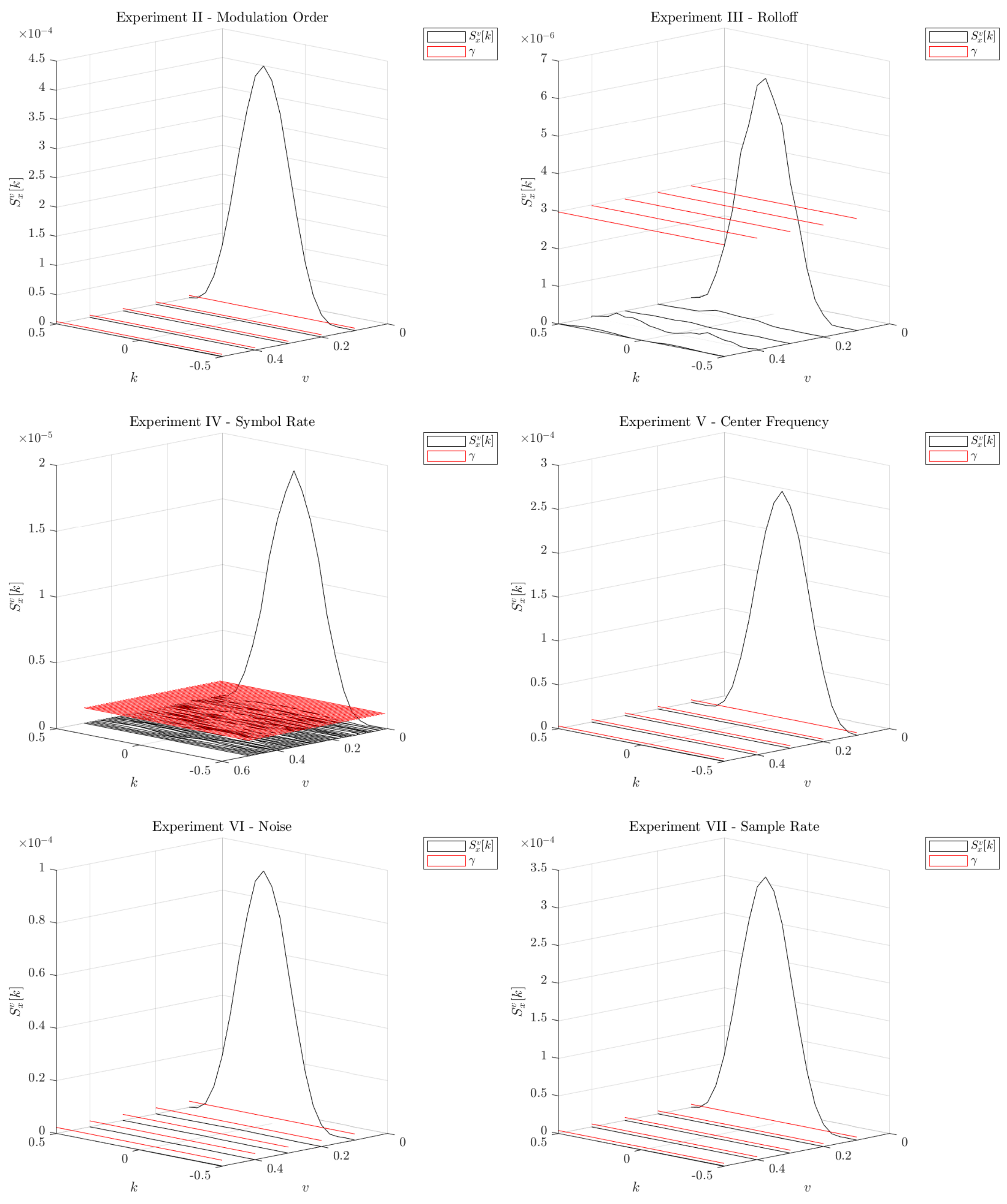


Figure 36: CSD estimate plots for experiments II-VII

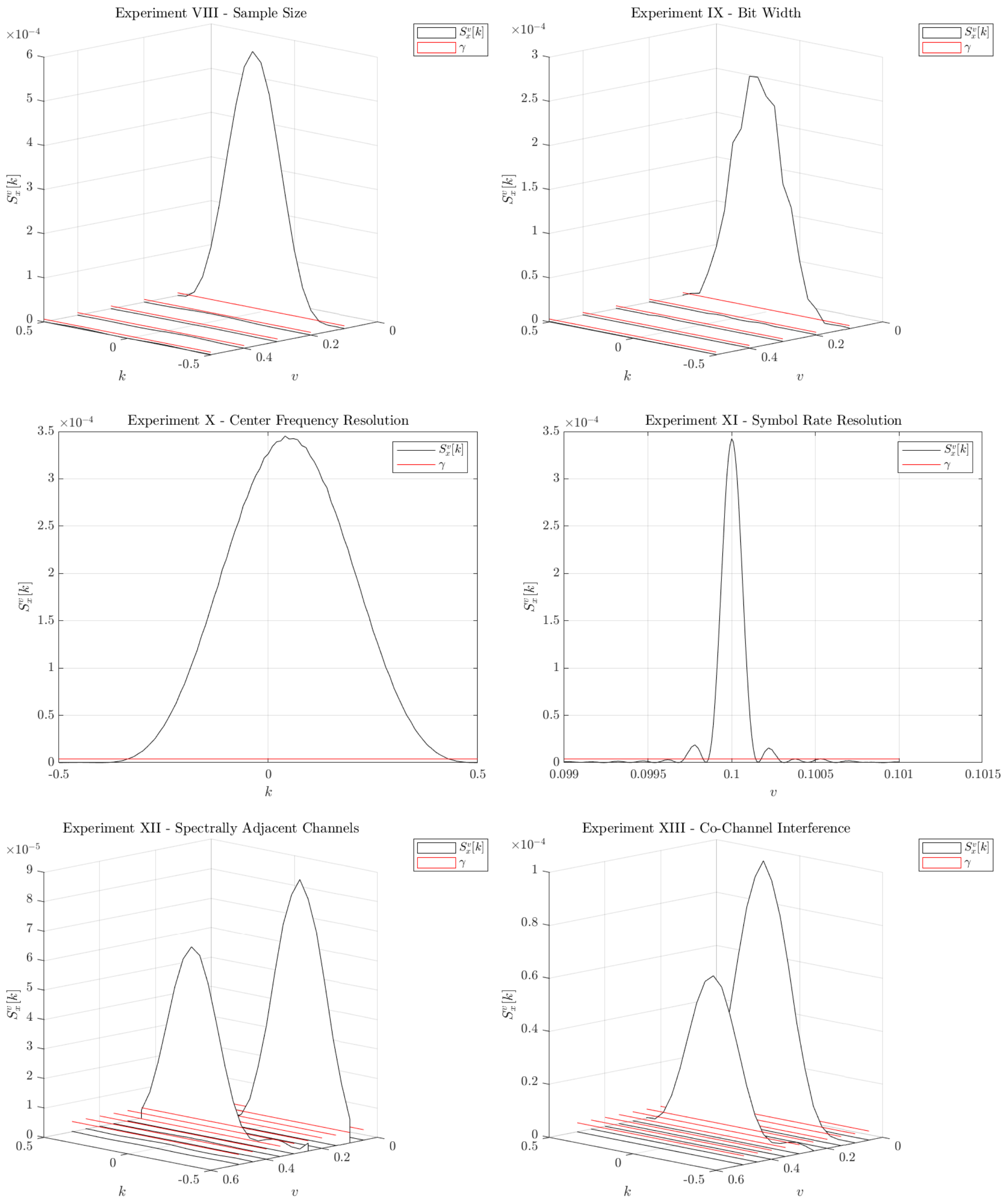


Figure 37: CSD estimate plots for experiments VIII-XIII

4.2 Results

From our experimentation, we were able to produce the results shown in Table 12 and Table 13. We find that every test surpasses the threshold and is successfully detected.

Table 12: Performance Results
Here, $\beta = 10^{-4}$. (R_s, f_c) are detected waveforms.

Expt.	$\boldsymbol{Delay\,(ns)}$	$\boldsymbol{\sigma_\eta^4}$	$\boldsymbol{R_s\,(MBd)}$	$\boldsymbol{f_c\,(MHz)}$
I	6881440500	0.0390605536703082500	0.1	0.05
II	6881440500	0.0398833126975169200	0.1	0.05
III	6881440500	0.0298027449329316400	0.1	0.05
IV	6881440500	0.0117707270580887540	0.1	0.05
V	6881440500	0.0296904514781464350	0.1	−0.05
VI	6881440500	0.0230728422450306340	0.1	0.05
VII	6881440500	0.0394264003727182250	0.1	0.05
VIII	430240500	0.0613674210127828100	0.1	0.05
IX	6881440500	0.0323822252535604900	0.1	0.05
X	6619200500	0.0390605536703082500	0.1	0.04
XI	13243153500	0.0390605536703082500	0.1	0.05
XII	13762840500	0.0353091800682484900	0.1; 0.25	−0.2; 0.2
XIII	13762840500	0.034193338983871176	0.1; 0.25	0.05; 0.25

Table 13: Resource Utilization Results

Expt.	***LUT***	***LUTRAM***	***FF***	***BRAM***	***DSP***	***IO***
I	814	3	391	15	16	108
II	814	3	391	15	16	108
III	814	3	391	15	16	108
IV	814	3	391	15	16	108
V	814	3	391	15	16	108
VI	814	3	391	15	16	108
VII	814	3	391	15	16	108
VII	814	3	391	15	16	108
IX	674	3	319	15	16	100
X	814	3	391	15	16	108
XI	814	3	391	15	16	108
XII	814	3	391	15	16	108
XIII	814	3	391	15	16	108

CHAPTER V
CONCLUSION

We investigated a low-memory FPGA implementation of an SCA for M-QAM-SRRC SR-CF detection. We began by developing a set of assumptions that limited the scope of existing SCA paradigms. We then discussed the HDL design used to construct such a system, which did not rely on FFTs and used no proprietary tools. Following this, we presented the performance characterization results of the newly formed streaming-SCA.

As discussed in section 3.2, one of the shortcomings of this design is that performance scales poorly with higher resolutions when compared to other methods. This difficulty in scaling is the main disadvantage when compared to the SSCA. Instead of blindly sweeping across all SRs with a high resolution, Koch instead uses a preprocessor to turn the SCA from blind to semi-blind. The concept is to use an FFT to produce the CAF then a threshold is applied. Values that meet the threshold are known as SR candidates fed into the streaming SCA to evaluate. This preprocessing can reduce the detection from thousands of (R_s, f_c) pairs to hundreds or even dozens. This thesis does not cover preprocessing due to the memory requirements of the FFT. However, preprocessor use is highly encouraged if the platform can afford the extra memory.

Another way to increase performance is through decimation. While this thesis does not cover the algorithm to sweep through (R_s, f_c) values, the general data flow is to determine the symbol rate to be checked, then decimate by $^{F_s}/_{R_s}$. Potential difficulties may be keeping track of bin's mapping to actual values and scheduling issues when using multiple streaming SCAs in parallel.

The threshold used in this algorithm is a basic example. This example threshold has several unsolved problems. The primary issue for some R_s, only one f_c can be detected by finding the maxima of a given spectral density. To solve this issue, an algorithm to detect local CF maxima above the threshold is needed. This enhancement would allow for the ability to detect multiple CFs sharing the same SR. The second issue is that when very fine values of α are used, there will be false detection for values close to the true SR. To solve this issue, an algorithm to detect local SR maxima above the threshold is needed. This enhancement would decrease the probability of SR false alarms. For now, peak detection is the best solution, but a study on the optimal detection for digital circuits would greatly expand this design's utility.

Lastly, M-QAM-SRRC is simply the easiest waveform to detect. Other PSFs, such as rectangular, can be support, in addition to other waveforms such as FSK, OOK, MSK, DSSS, DPSK, and encodings such as Manchester, are all supported by the streaming-SCA. However, optimal detection is much more difficult on these waveforms, and are not covered in this thesis.

REFERENCES

[1] C. M. Spooner. *Cyclostationary Signal Processing.* PhD thesis, University of California, Davis, David, California, June 1992.

[2] W. A. Gardner, A. Napolitano, and L. Paura. Cyclostationary: Half a century of research. *Signal Processing*, 86(4):639–697, 2006.

[3] W. A. Gardner. *Introduction to Random Processes with Applications to Signals and Systems.* McGraw-Hill, New York City, New York, 1990.

[4] M. V. Koch and J. A. Downey. Interference mitigation using cyclic autocorrelation and multi-objective optimization. Technical Report NASA/TM–2019-220226, National Aeronautics and Space Administration, Glenn Research Center, Cleveland, Ohio, July 2019.

[5] I. Ilyas, S. Paul, A. Rahman, and R. K. Kundu. Comparative evaluation of cyclostationary detection based cognitive spectrum sensing. In *Proceedings of the 7th Annual Ubiquitous Computing, Electronics, and Mobile Communication Conference*, IEEE-UEMCON, pages 1–7, New York City, New York, 2016. IEEE.

[6] Y. C. Liang. *Spectrum Sensing Theories and Methods*, pages 44–81. Springer, Singapore, Singapore, 2020.

[7] H. L. Van Trees. *Classical Detection and Estimation Theory*, pages 19–45. Wiley-Interscience, Hoboken, NJ, 2001.

[8] S. Bandyopadhyay, G. Subramanian, R. Foust, D. Morgan, Chung, and F. Hadaegh. A review of impending small satellite formation flying missions. In *Proceedings of the 53rd AIAA Aerospace Sciences Meeting*, AIAA-ASM, pages 1–17, Kissimmee, Florida, 2015. ARC.

APPENDIX A
SOURCE CODE

```
%% RX RF Frontend
norm_factor = modnorm(syms_noisy,'peakpow',1);
agc = syms_noisy*norm_factor;

%% Receiver: SCA
Ncapture = length(agc);

% valid alpha range is [0 Fs/2]
a_step = Rs; % only evaluate harmonics of Rs
a_array = (Rs:a_step:Fs/2)';
Na = length(a_array);

% valid fc range is [-Fs/2 Fs/2]
fc_step = fc; % only evaluate harmonics of fc
fc_array = (-Fs/2:fc_step:Fs/2)';
Nfc = length(fc_array);

% two tap low-pass filter
hlpf = [0.5,0.5];

% allocate memory for results
S = zeros(Na,Nfc);

% search cycle freqs
for idx_a = 1:Na
```

```
  % search center freqs
  for idx_fc = 1:Nfc
    % channelizer
    lo = exp(1j*2*pi*fc_array(idx_fc)/Fs*(1:Ncapture))';
    channelized = filter(hlpf,1,syms_pb(1:Ncapture).*lo);

    % symbol rate estimation
    R = abs(channelized).^2;
    S(idx_a,idx_fc) =
        goertzel(R,round(a_array(idx_a)/Fs*Ncapture)+1)/Ncapture;
  end
end

T = abs(S).^2;
Pfa = 1e-3;
sigma = Pfa*(sum(abs(agc).^2)/Ncapture)^2;
threshold = ones(Na,Nfc)*sigma
```

APPENDIX B
ACRONYMS

ADC	analog to digital converter
AF	autocorrelation function
AWGN	additive white Gaussian noise
BRAM	block random access memory
CAF	cyclic autocorrelation function
CF	center frequency
CFD	cyclostationary feature detector
CFE	center frequency estimator
CPU	central processing unit
CR	cognitive radio
CSD	cyclic spectral density
CSP	cyclostationary signal processing
CubeSat	cube satellite
DC	direct current
DFT	discrete Fourier transform
DSP	digital signal processor
DVB-S2	digital video broadcasting - satellite generation two
ESD	energy spectral density
FAM	FFT accumulation method
FCW	frequency control word
FFT	fast Fourier transform
FIR	finite impulse response
FPGA	field-programmable gate array
FAM	FFT accumulation method
FSM	FFT accumulation method
GPU	graphics processing unit
HDL	hardware description language
HSDPA	high speed downlink packet access
IID	independent and identically distributed
IQ	in-phase and quadrature
LPF	low-pass filter
M	modulation order
MATLAB	matrix laboratory
NCO	numerically-controlled oscillator
PDF	probability density function
PM	pulse modulation
PSD	power spectral density
PSF	pulse-shape filter
PU	primary user
PSK	phase-shift keying
QAM	quadrature amplitude modulation

RF	radio frequency
SCA	spectral correlation analyzer
SNR	signal to noise ratio
SOC	second-order cyclostationary
SOS	second-order stationary
SR	symbol rate
SRE	symbol rate estimator
SRRC	square-root-raised-cosine
SSCA	split spectral correlation analyzer
SU	secondary user
SWaP	size, weight, and power
TSM	time-smoothing method
UMTS	universal mobile telecommunications system
VDHL	VHSIC-HDL
VHSIC	very high speed integrated circuit
WSC	wide-sense cyclostationary
WiMAX	worldwide interoperability for microwave access
WSS	wide-sense stationary