Software-Defined Radio Implementation of Two Physical Layer Security Techniques

Kevin S. Ryland

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

Thomas C. Clancy
Richard M. Buehrer
Carl B. Dietrich

December 19th, 2017
Arlington, Virginia

Keywords: Physical Layer Security, Software Defined Radio, Alamouti STBC, Artificial Noise Generation, Over-the-Air

Software-Defined Radio Implementation of Two Physical Layer Security Techniques

Kevin S. Ryland

ABSTRACT

This thesis discusses the design of two Physical Layer Security (PLS) techniques on Software Defined Radios (SDRs). PLS is a classification of security methods that take advantage of physical properties in the waveform or channel to secure communication. These schemes can be used to directly obfuscate the signal from eavesdroppers, or even generate secret keys for traditional encryption methods. Over the past decade, advancements in Multiple-Input Multiple-Output systems have expanded the potential capabilities of PLS while the development of technologies such as the Internet of Things has provided new applications. As a result, this field has become heavily researched, but is still lacking implementations. The design work in this thesis attempts to alleviate this problem by establishing SDR designs geared towards Over-the-Air experimentation.

The first design involves a 2×1 Multiple-Input Single-Output system where the transmitter uses Channel State Information from the intended receiver to inject Artificial Noise (AN) into the receiver's nullspace. The AN is consequently not seen by the intended receiver, however, it will interfere with eavesdroppers experiencing independent channel fading. The second design involves a single-carrier Alamouti coding system with pseudo-random phase shifts applied to each transmit antenna, referred to as Phase-Enciphered Alamouti Coding (PEAC). The intended receiver has knowledge of the pseudo-random sequence and can undo these phase shifts when performing the Alamouti equalization, while an eavesdropper without knowledge of the sequence will be unable to decode the signal.

Software-Defined Radio Implementation of Two Physical Layer Security Techniques

Kevin S. Ryland

ABSTRACT FOR A GENERAL AUDIENCE

This thesis discusses the design of two Physical Layer Security (PLS) techniques. PLS is a classification of wireless communication security methods that take advantage of physical properties in transmission or environment to secure communication. These schemes can be used to directly obfuscate the signal from eavesdroppers, or even generate secret keys for traditional encryption methods. Over the past decade, advancements in Multiple-Input Multiple-Output systems have expanded the potential capabilities of PLS while the development of technologies such as the Internet of Things has provided new applications. As a result, this field has become heavily researched, but is still lacking implementations. The design work in this thesis attempts to alleviate this problem by establishing systems that can be used for laboratory experimentation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Confidentiality in modern communication systems is constantly challenged by the broadcast nature of the wireless medium. This problem is traditionally solved by encrypting the message at upper layers of the network stack. As new technology appears, new applications emerge that present fundamental problems for traditional encryption. One example is that with the increasing number of power-constrained and computationally-limited devices being incorporated into Internet-of-Things (IoT) networks, lightweight security methods have become essential [10].

Physical Layer Security (PLS) is a classification of security methods that take advantage of physical properties in the waveform or channel to secure communication. PLS, as it applied to confidentiality, is broken up into two main branches. The first branch deals with directly obfuscating the transmission from an eavesdropper. These techniques attempt to degrade the reception for eavesdroppers while simultaneously ensuring that the message is successfully communicated to the intended recipient. This is accomplished by exploiting the unique properties of the channel shared by the transmitter and intended receiver. The second branch of PLS focuses on using the unique characteristics of the channel between the transmitter and intended receiver to generate a private key that can be used to encrypt communication.

Wyner laid the foundation for PLS in 1975 by introducing the concept of information-theoretic security in the context of a discrete memoryless wiretap channel modeling the communication of two legitimate parties in the presence of an eavesdropper [1]. Figure 1.1 shows the general case of the Wyner wiretap channel where legitimate users communicate over a main channel and are observed by an eavesdropper through an additional wiretap channel.

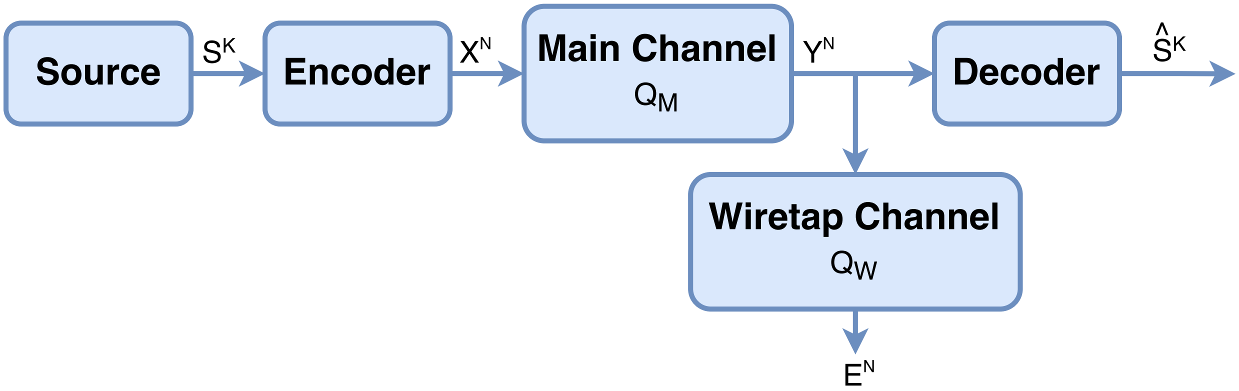


Figure 1.1: General Case of the Wyner Wiretap Channel [1].

Wyner's work proved that when the intended receiver operates in a more favorable channel than the eavesdropper, there is a quantifiable amount of information that can be communicated in perfect secrecy. This is the fundamental concept behind the obfuscation branch of PLS techniques.

The key-generation branch relies on the transmitter and intended receiver mutually measuring a unique property of their shared channel. Common properties include the Received Signal Strength Indicator (RSSI), complex channel coefficients, and channel phase [11]. A practical consideration when measuring these properties on both ends of the channel is the accuracy of the measurement and the potential correlation of the measurement with the eavesdropper. In the case of measuring RSSI, the system will round the result to ensure both the transmitter and receiver are measuring the same value. Additionally, the designer may want to remove one or more of the most significant figures of the measurement if the eavesdropper is likely to measure the same value.

The adoption of Multiple-Input Multiple-Output (MIMO) communication in 802.11n and LTE created a resurgence in PLS research over the past decade. MIMO provides more opportunities for enhancing security at the physical layer such as adaptively steering a null towards an eavesdropper or simply beamforming in the direction of the intended receiver. The drawback is that eavesdroppers can now take advantage of multiple antennas in a MIMO Multiple Eavesdropper (MIMOME) channel to reduce the effectiveness of these techniques [12].

Information-theoretic security is claimed to be a stricter level of security than traditional encryption methods [11]. The rationale behind this assertion is that cryptographic encryption is built on the assumption that it will be computationally infeasible for an eavesdropper to decrypt the ciphertext without the secret key. This assumption is not mathematically rigorous and there are examples of ciphers being broken due to a combination of flaws in implementation and technological advancements. A recent example is the generation of a collision for the SHA-1 hashing function [13]. Information theoretic security can provide provably perfect security at data rates under the secrecy rate of a wiretap channel. The catch is that to reliably measure the secrecy rate, the channel to all eavesdroppers must be known. In the case of a passive eavesdropper, only probabilistic secrecy measures can be obtained.

Hidden issues with the implementation of cryptographic algorithms can also undermine security against unsophisticated attacks. A great example of this occurred in Debian's OpenSSL in September of 2006 when a software patch restricted the psuedo-random number generator to being seeded only by the process ID, creating keys with only 32,767 unique values on an individual architecture [14]. Across the 3 possible architectures (i386, amd64, and ia64), this resulted in only $(2^{15} - 1) * 3 = 98301$ unique keys. Debian was using a 1024-bit DSA key at the time, which was believed to take on 2^{1024} values! What's even more disturbing is that this vulnerability was not found or reported until May 2008, almost 2 years later [15]. With the integrity of a cryptographic algorithm buried deep in the source code of the implementation, new security vulnerabilities are being found every day. Therefore, it is imperative that more robust security methods are developed just as quickly.

Cryptographic techniques operate independently from the physical layer, making PLS an easy option to augment existing security systems. PLS techniques can be practically applied in conjunction with traditional encryption or to provide lightweight security solutions for massive networks.

This thesis discusses the design of two PLS techniques in GNU Radio to facilitate Over-the-Air (OTA) experimentation. The first design involves a 2×1 Multiple-Input Single-Output (MISO) system where the transmitter uses Channel State Information (CSI) from the intended receiver to inject Artificial Noise (AN) into the receiver's nullspace. The AN is consequently not seen by the intended receiver, however, it will interfere with eavesdroppers in an independent channel realization. The second design involves a single-carrier Alamouti coding system with pseudo-random phase shifts applied to each transmit antenna, referred to as Phase-Enciphered Alamouti Coding (PEAC). The intended receiver has knowledge of the pseudo-random sequence and can undo these phase shifts when performing the Alamouti equalization, while an eavesdropper without knowledge of the sequence will be unable to decode the signal.

Chapter 2

Background

PLS is a diverse research area drawing from concepts in Information Theory and MIMO communications systems. Furthermore, an understanding of Digital Signal Processing (DSP) and Software-Defined Radio (SDR) is necessary for implementing PLS techniques. This section aims to cover the necessary background knowledge in each of these area in order to successfully understand and design a PLS system.

2.1 Information-Theoretic Security

2.1.1 Shannon Channel Capacity

Channel Capacity in Terms of Error Correction Coding

Communication channels can be characterized by a set of transition probabilities. As an example, consider the Binary Symmetric Channel (BSC) in Figure 2.1. For an input of 0, the probability of a bit flip is f and the probability of a successful transmission is $1 - f$. The same transition probabilities hold true for an input of 1, this is why it is called a symmetric channel. Over a large number of bits, the Bit Error Rate (BER) of a BSC is approximately f.

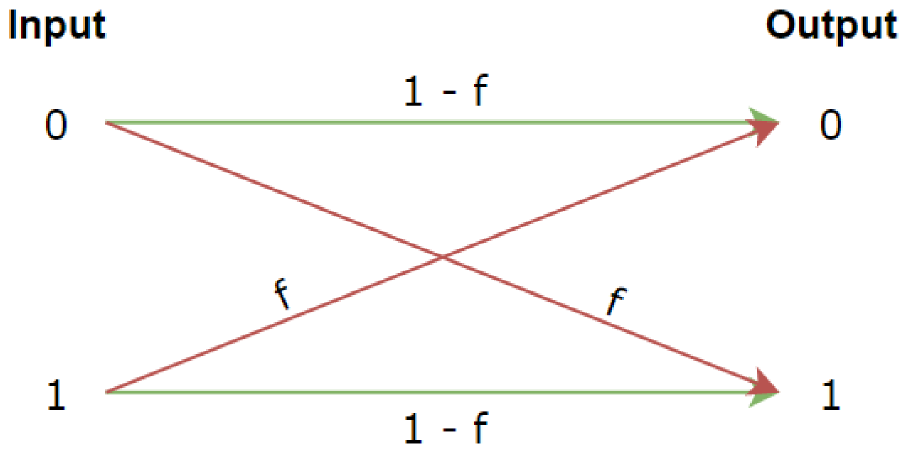


Figure 2.1: The Binary Symmetric Channel with flip probability f.

The flip probability of the BSC is tied to some physical aspect of the channel which often cannot be modified enough to reduce the BER to an acceptable level. For example, if a hard disk drive is modeled as a BSC, then the flip probability is a function of the physical components such as the connectors, platter, and reader. It is not realistic to rely solely on improving these components in order to reach a target BER. Therefore, the system must be designed to accept errors by detecting and correcting them. This is accomplished through error correction coding.

Consider the coding system below in Figure 2.2.

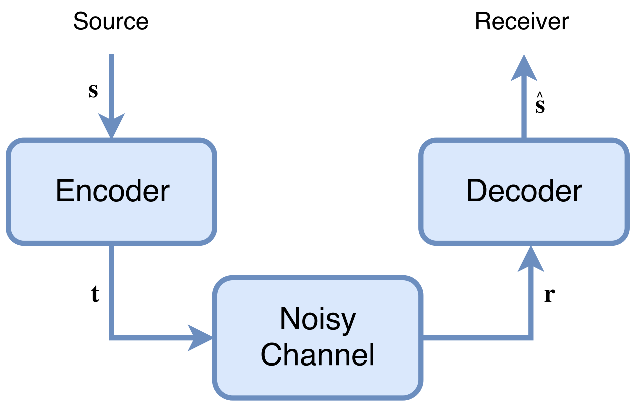


Figure 2.2: Error correction coding system [2].

First, the source bits are run through an encoder at the transmitter. The encoder attempts to protect against potential errors by adding redundancy to the source bits. Next, the encoded bits are sent through a noisy channel and experience some errors before being received. The receiver then decodes the bits, removing the redundancy added at the encoder while correcting for errors. The output of the decoder is an estimate of the source bits.

The amount of redundancy added by a particular coding scheme is characterized by the coding rate, $R = K/N$, where K is the number of source bits in the code and N is the total code length. The coding rate is a measure of the spectral efficiency of a code, indicating the amount of redundancy the code adds to the transmitted information.

Each coding scheme can be characterized by its coding rate and the BER that the scheme achieves in a particular channel. For a BSC with a flip probability of $f = 0.1$, the rate-BER pairs for several coding schemes are shown in Figure 2.3 [2].

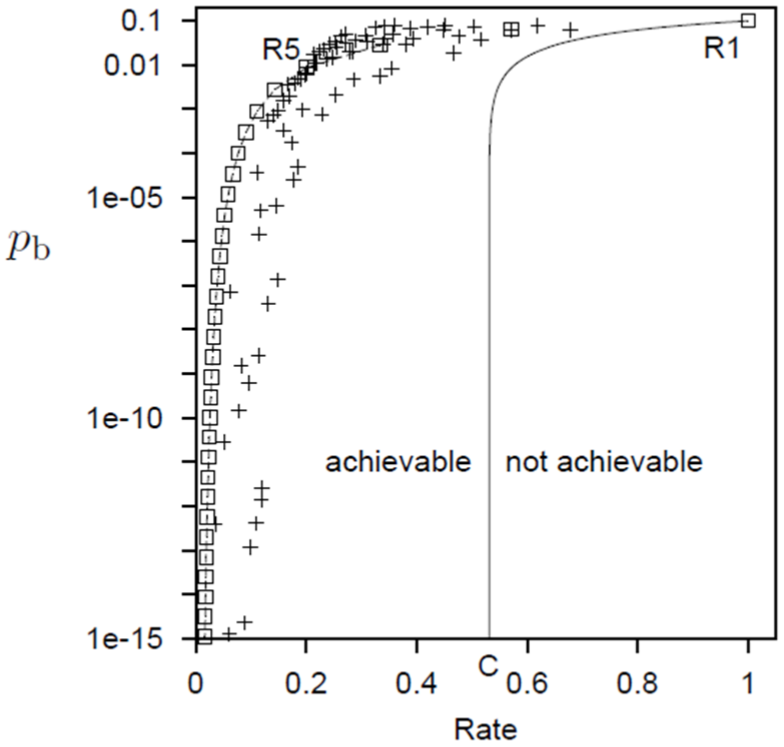


Figure 2.3: BER vs. Code Rate Graph of popular error-correction codes [2].

As illustrated in Figure 2.3, the general trend observed for error correction codes is that as the BER achieved by a code decreases, the coding rate approaches 0. This is why it was traditionally thought that to achieve an infinity small BER, the coding rate must be 0.

Shannon demonstrated that there can be a non-zero upper-limit to the rate of the channel, under which an arbitrarily small BER can be achieved. This upper-bound is termed the channel capacity or Shannon capacity of a channel. The channel capacity is illustrated in Figure 2.3 as a point on the x-axis which defines a boundary between the achievable and non-achievable regions in the BER-Code Rate space. The idea that an infinitely small BER can be achieved for some non-zero (and in this example, significantly large) coding rate is non-intuitive from the results. The trade-off here is that while the code rate can be kept at some non-zero value below or equal to the channel capacity, the code length must increase to achieve lower BERs for a given code. This results in codes that are impractical because of large block size and the resulting latency that is introduced.

Channel Capacity in Terms of Information Measures

The Shannon information content of an outcome, x, with probability, p, is

$$h(p) = \log_2 \left(\frac{1}{p}\right).$$

An ensemble, X is defined as $X \in \{x, A_X, P_X\}$ where x is a set of outcomes, A_X is the set of possible values the outcomes can take on, and P_X is a set of the corresponding probabilities of each value in A_X.

The Entropy of the ensemble, X, is

$$H(X) = \sum_{x \in A_X} P(x)h(x).$$

Joint, conditional, and marginal entropies can be related through the entropy chain rule which is depicted in Figure 2.4.

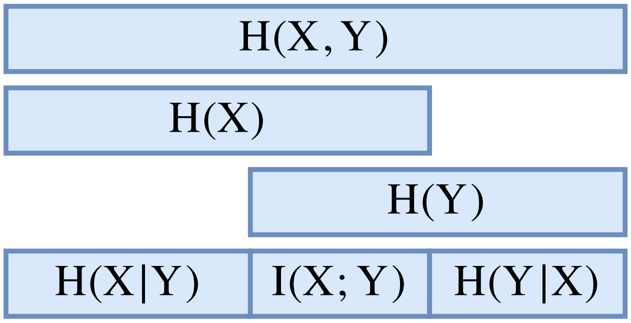


Figure 2.4: Entropy Chain Rule, derived from [2].

The entropy chain rule relations introduce a new information measure, the mutual information $I(X;Y) = I(Y;X)$, defined as

$$I(X;Y) = H(X) - H(X|Y)$$

$$I(Y;X) = H(Y) - H(Y|X).$$

The mutual information is a measure of the information Y conveys about X or vice versa, although the former definition will be used more often in a communications context.

The channel capacity is defined as the maximum mutual information obtained by optimizing the input distribution to the channel

$$C = I_{max(P_x)}(X;Y).$$

2.1.2 Wyner Wiretap Channel

Special Case

In 1975, Aaron Wyner introduced the wiretap channel which models the communication of two legitimate communicants in the presence of an eavesdropper. Figure 2.5 shows a special case of this wiretap channel where the legitimate parties communicate over a noiseless channel and are observed by an eavesdropper through a Binary Symmetric Channel [1]. The encoder operates on blocks of K source bits $S^K = (S_1, S_2, ..., S_K)$ and produces an encoded sequence $X^N = (X_1, X_2, ..., X_N)$ of length N.

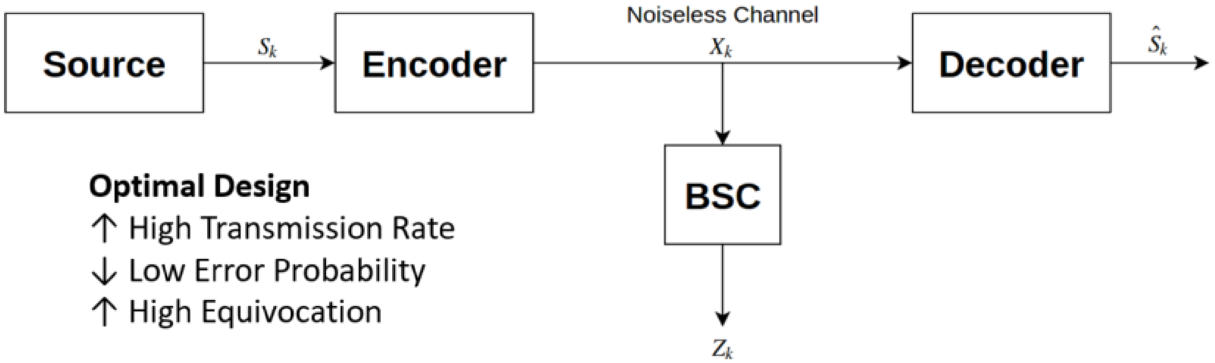


Figure 2.5: Special Case of the Wyner Wiretap Channel.

There are three design metrics in Wyner’s wiretap channel - transmission rate, error probability, and equivocation rate. The transmission rate of the channel is defined by the ratio of the information bits sent to the total code length

$$R = K/N.$$

The error probability is defined as

$$P_e = \frac{1}{K}\sum_{k=1}^{K}[S_k \neq \hat{S}_k].$$

The eavesdropper observes the encoded sequence X^N through a discrete memoryless BSC termed the wiretap channel which has a flip probability f as depicted in Figure 2.1. The output of the wiretap channel $Z_N = (Z_1, Z_2, ..., Z_N)$ is observed by the eavesdropper.

The equivocation rate

$$\Delta = \frac{1}{K}H(S^K|Z^N)$$

is a measure of the confusion experienced by the eavesdropper.

As indicated by Figure 2.5, this channel is designed to have a high transmission rate, a high equivocation rate, and a low error probability.

Wyner illustrates the relationship between transmission rate and equivocation by examining two examples of the special case wiretap channel.

(Example 1) Let $K = N = 1$ and $X_1 \equiv S_1$

This results in $P_e = 0$, $R = K/N = 1$, and $\Delta = h_2(f)$ where

$$h_2(\lambda) = \lambda \log_2(\frac{1}{\lambda}) + (1-\lambda)\log_2(\frac{1}{1-\lambda})$$

is the binary entropy function.

(Example 2) Let $K = 1$, N is arbitrary,

$$C_i = \begin{cases} \{0,1\}^N \text{with even parity} & \text{if } i = 0 \\ \{0,1\}^N \text{with odd parity} & \text{if } i = 1 \end{cases}$$

and X^N chooses a random sequence from C_i for $S_1 = i$.

For $z \in \{0,1\}^N$ with even parity,

$$P(S_1 = 0|Z^N = z) = P(\text{even number of errors in BSC})$$

$$P(S_1 = 0|Z^N = z) = \sum_{j=0,even}^{N} f^j (1-f)^{N-j} = \frac{1}{2} + \frac{1}{2}(1-2f)^N$$

For $z \in \{0,1\}^N$ with odd parity,

$$P(S_1 = 0|Z^N = z) = P(\text{odd number of errors in BSC})$$

$$P(S_1 = 0|Z^N = z) = \sum_{j=0,odd}^{N} f^j (1-f)^{N-j} = \frac{1}{2} - \frac{1}{2}(1-2f)^N$$

For all $z \in \{0,1\}^N$,

$$H(S_1|Z^N = z) = \Delta = h_2(\frac{1}{2} - \frac{1}{2}(1-2f)^N)$$

Notice that as $N \to \infty$, $\Delta \to 1 = H(S_1)$. Therefore, as $N \to \infty$, the communication is accomplished in perfect secrecy, but the transmission rate $R = \frac{1}{N} \to 0$.

General Case

In the general case of the Wyner wiretap channel shown in Figure 2.6, the main and wiretap channels are discrete, memoryless, and characterized by their transition probabilities, Q_M and Q_W, respectively. The probability of bit error and equivocation measures for the general wiretap channel remain unchanged from the special case. The source, S^K, is now defined as a sequence of independent and identically distributed (IID) random variables that can take on values from an arbitrary alphabet, S. As a results, the transmission rate now incorporates the source entropy, H_S,

$$R = \frac{KH_s}{N}.$$

The error probability and equivocation are measured identically to the special case.

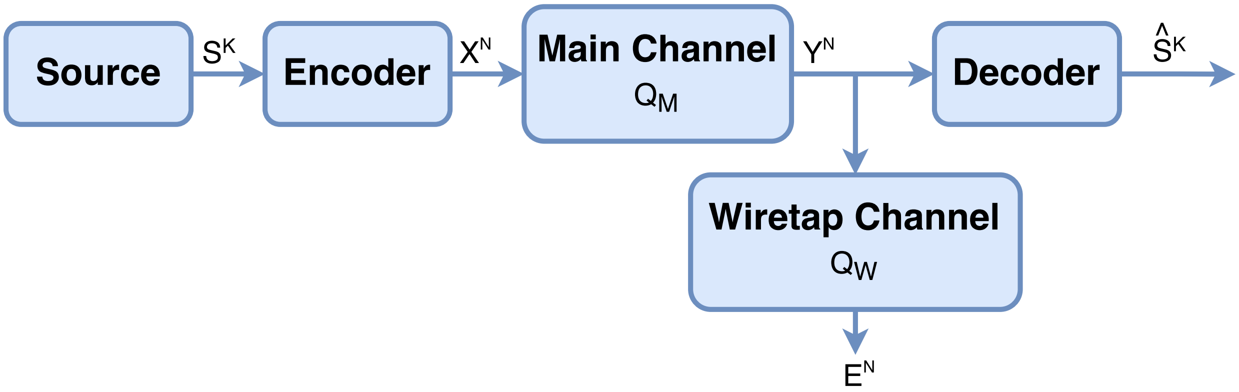


Figure 2.6: General Case of the Wyner Wiretap Channel [1].

Secrecy Capacity Region

Consider the transmission-equivocation rate pair, (R, d). For the first coding example pertaining to the special case wiretap channel, the pair $(1, h_2(f))$ is achieved. In the second example coding scheme, the pair $(0, H_S)$ is achieved. This is analogous to the coding rate - BER pairs that make up Shannon's Channel Capacity region in Figure 2.3. Each coding scheme achieves a particular pair in a given channel which can be compared in terms of error performance vs. spectral efficiency. Wyner now compares the pairs in terms of achieved secrecy vs. spectral efficiency.

Wyner characterizes a region of achievable rate-equivocation pairs **R** shown in Figure 2.7.

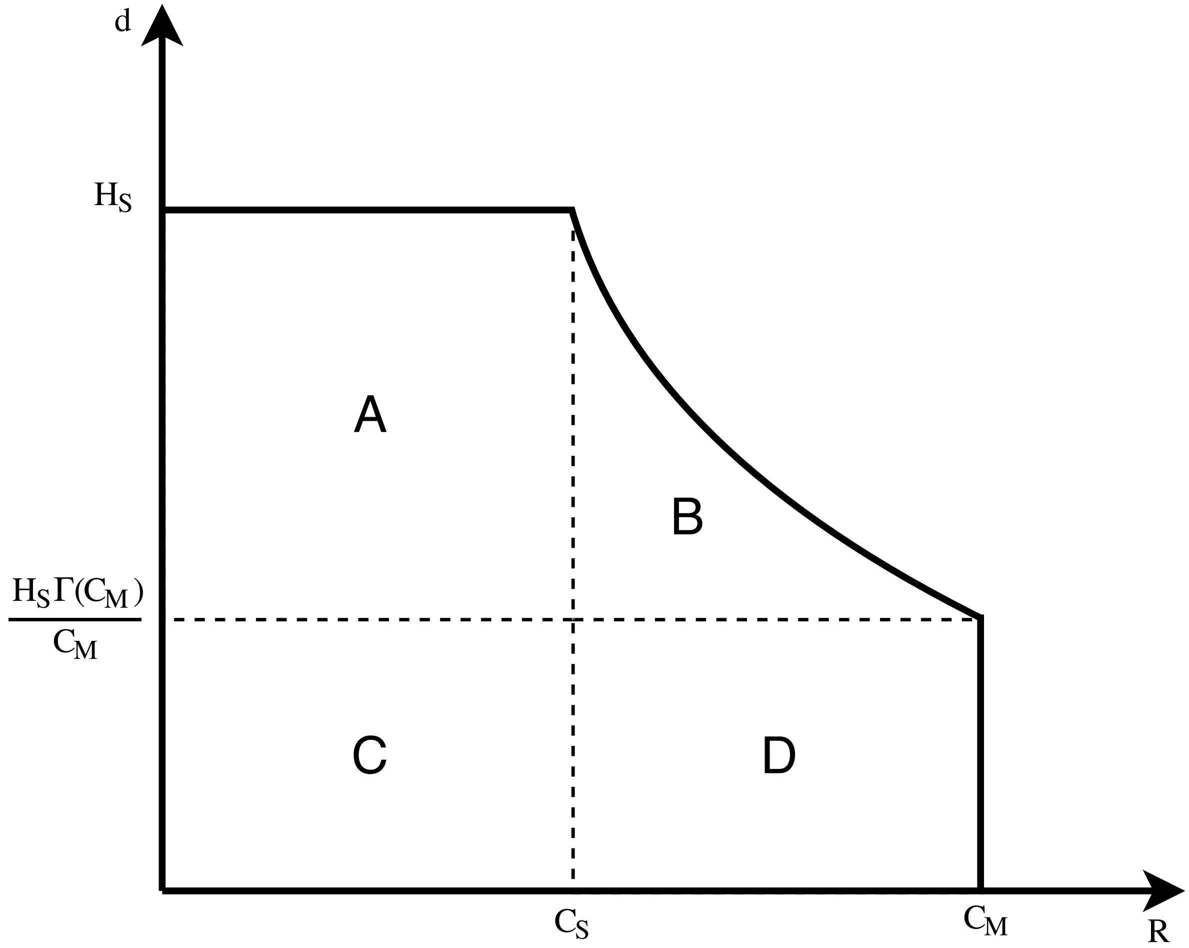


Figure 2.7: Region of achievable Wyner wiretap codes (A,B,C, and D) [1].

The region of achievable pairs, $\bar{R}$, shown in Figure 2.7 is defined by

$$\bar{\mathbf{R}} = \{(R,d) : 0 \leq R \leq C_M, 0 \leq a \leq H_S, Rd \leq H_S\Gamma(R)\}$$

The function $\Gamma(R)$ is used to rigorously define the maximum difference in mutual information that is possible by optimizing the input distribution

$$\Gamma(\mathbf{R}) = \sup_{p \in P}(I(X;Y) - I(X;Z)).$$

In security, it is common nomenclature to refer to the transmitter as Alice, the intended receiver as Bob, and the eavesdropper as Eve. For the remainder of the thesis, this is how each of these participants will be named.

2.1.3 Example Wiretap Code

In this section, an example wiretap code is explored to give the reader a better understanding of information-theoretic security concepts.

Consider the scenario where Bob has a SNR high enough to successfully demodulate 16-QAM while Eve's SNR is limited (through mechanisms discussed later on) such that she can only demodulate QPSK. Bob's received constellation, depicting a potential wiretap coding scheme for this scenario, is shown in Figure 2.8.

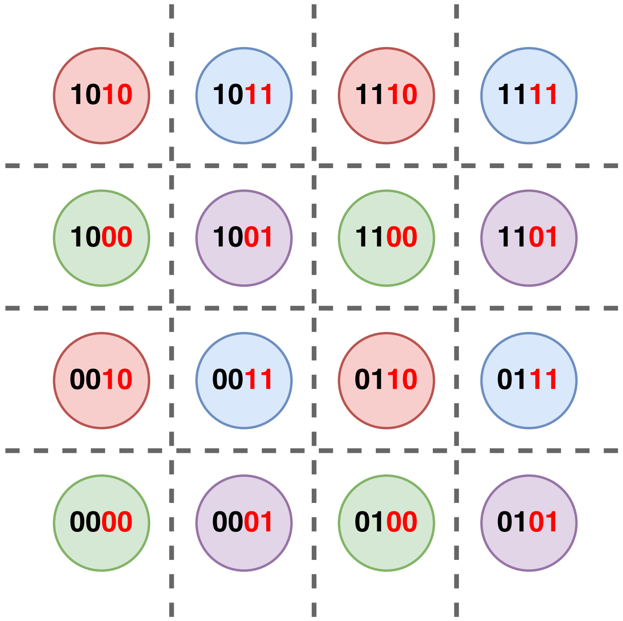


Figure 2.8: Example wiretap code applied to a 16-QAM modulated signal.

The bit mapping for each symbol contains red "protected bits" and black "unprotected bits." The assertion is that due to the capacity difference present in this scenario, Bob can obscure the protected bits from Eve using this wiretap code. Next, consider Eve's Error Vector Magnitude (EVM) in comparison to Bob's in Figures 2.9 and 2.10.

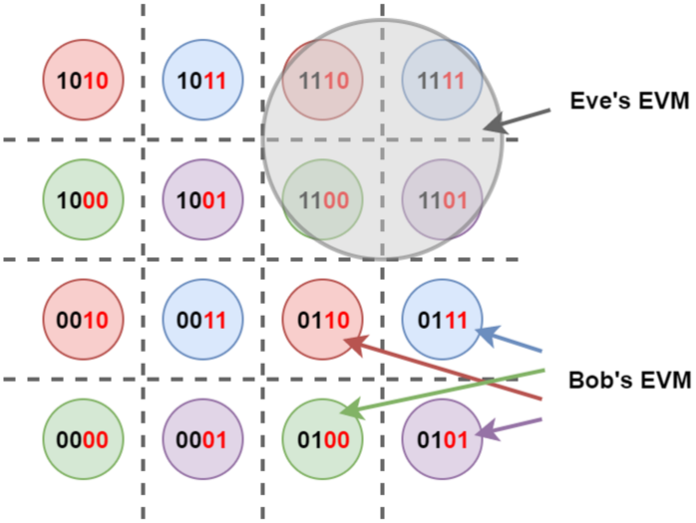


Figure 2.9: Comparison of Bob's and Eve's EVMs.

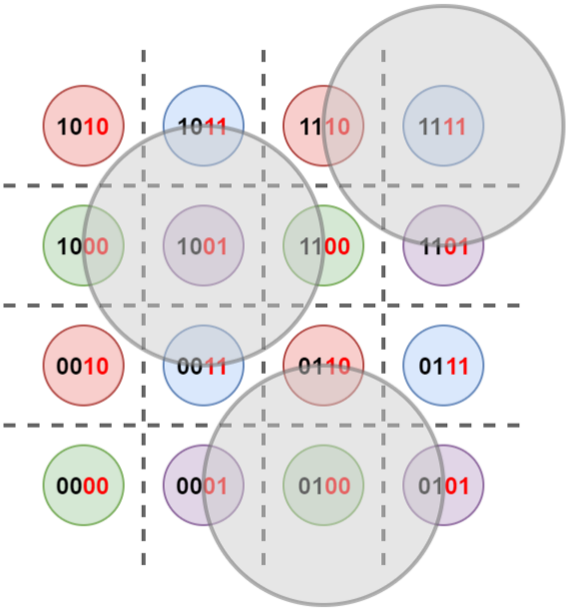


Figure 2.10: Possible overlap scenarios between Bob and Eve's EVMs.

Eve's EVM is chosen to be as large as possible while still being able to demodulate QPSK with no errors. Figure 2.9 shows that Eve's EVM fits exactly in a single quadrant to satisfy this requirement. Figure 2.10 compares Bob's and Eve's EVMs in three unique cases where both EVMs are centered on different received symbols: a corner symbol (blue), a central symbol (purple), and an edge symbol (green). Figure 2.11 examines the case involving the central symbol.

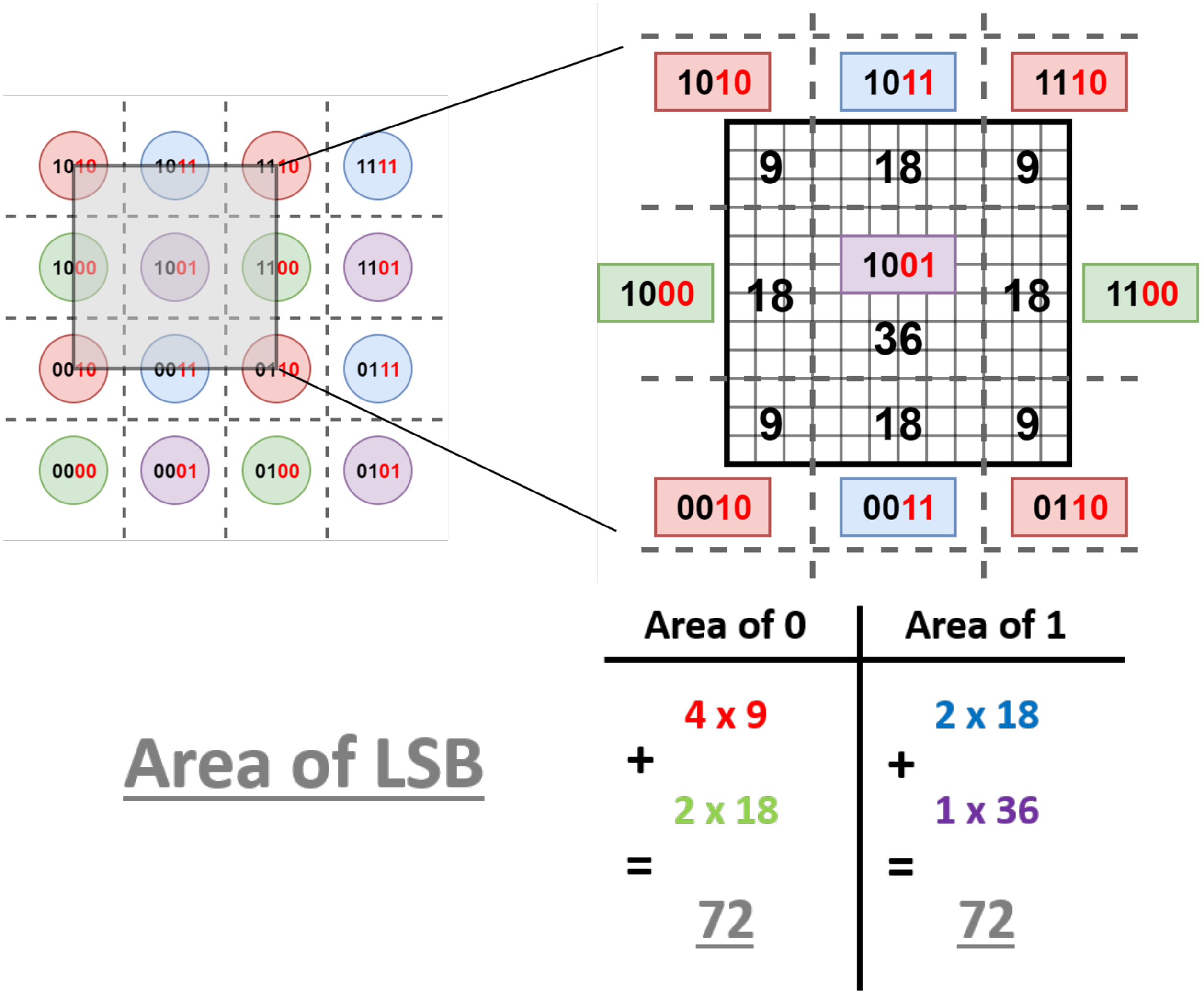


Figure 2.11: Analysis of Eve's confusion when decoding the LSB of a central symbol.

A bit is considered completely obscured if the areas that it can be present as either a 0 or 1 within Eve's EVM are equal. In other words, Eve should have a perfectly ambiguous choice between a 0 or 1 for the bit. An assumption that must be made for this analysis is that Eve's EVM is the entire quadrant, rather than a circle that spans the quadrant as pictured in Figure 2.9.

Figure 2.11 shows that the areas where the protected bits are decoded as a 0 and 1 are both equal within Eve's square EVM. This can be similarly demonstrated for the second to least significant bit. Additionally, this property will also hold for the remaining two cases.

This example emphasizes the utility of wiretap coding. Wiretap codes allows the system to focus the secrecy provided by a capacity gap into protection of particular data. It is important to distinguish that wiretap codes discipline the secrecy provided by physics, they do not produce security on their own.

2.2 Multiple-Input Multiple-Output Communications

Multiple-Input Multiple Output (MIMO) communications has become an extremely popular and well-researched field in the past 20 years. This is a result of the performance benefits provided by MIMO systems which can be categorized as array gain, diversity gain, multiplexing gain, and interference reduction [16]. Array gain is produced by coherently combining signals with channel knowledge. Diversity gain characterizes the increased resilience to fading due to having multiple receive elements with independent channel realizations. Multiplexing gain is the increase in capacity that comes from simultaneously transmitting multiple data streams and separating them based on the spatial characteristics of the communicating array elements. Interference reduction describes the ability of MIMO systems to minimize interference from co-channel users in communication infrastructures that rely on frequency reuse such as cellular systems. While not all of these benefits may be exploited simultaneously, they are individually valuable enough to inspire adoption in many modern wireless standards.

The focus of this thesis is not on any of the traditional capacity benefits that MIMO provides, but rather on how MIMO systems impact security. Since the secrecy capacity is directly proportional to the difference in capacities of the main and wiretap channels, the security benefits are still fundamentally tied to capacity benefits.

The channel model of a received signal in a single-user, flat-faded, MIMO channel model is

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n}$$

where $\mathbf{y}$ is the $M_R \times 1$ received signal vector, $\mathbf{x}$ is the $M_T \times 1$ transmitted signal vector, $\mathbf{n}$ is the $M_R \times 1$ noise vector, and $\mathbf{H}$ is the $M_R \times M_T$ channel matrix. A channel is said to be flat-faded when the fading effects of the channel on the transmitted signal are independent of frequency. This assumption is valid only when the transmitted signal is within the coherence bandwidth of the channel. The coherence bandwidth is an intrinsic property of the channel, determined by how it delays reflected components of the transmitted signal. In contrast, if the fading effects of the channel were frequency-dependent, then each element of $\mathbf{H}$ would be equivalent to a Finite-Impulse Response (FIR) filter. Flat-fading is an important simplifying assumption which will be used throughout this thesis.

The received signal model can be further simplified using the Singular Value Decomposition (SVD) to decompose the channel matrix H into parallel, non-interfering, SISO channels. The SVD of $\mathbf{H}$ is written as

$$\mathbf{H} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathbf{H}}$$

where $\mathbf{U}$ is a $M_R \times M_R$ unitary matrix, $\mathbf{V}$ is a $M_T \times M_T$ unitary matrix, and $\boldsymbol{\Sigma}$ is a $M_R \times M_T$ diagonal matrix. The $R_H = min(M_T, M_R)$ diagonal entries of $\boldsymbol{\Sigma}$ are called the singular values of $\mathbf{H}$ and are assumed to be in descending order such that $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_{R_H}$.

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1 & 0 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 & 0 \\ 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & 0 & \sigma_{R_H} & 0 \end{bmatrix}$$

MIMO Systems separate the channels by precoding the input signal $\mathbf{x} = \mathbf{Vs}$ and post-multiplying the output $\mathbf{y} = \mathbf{U^H r}$ as illustrated in Figure 2.12.

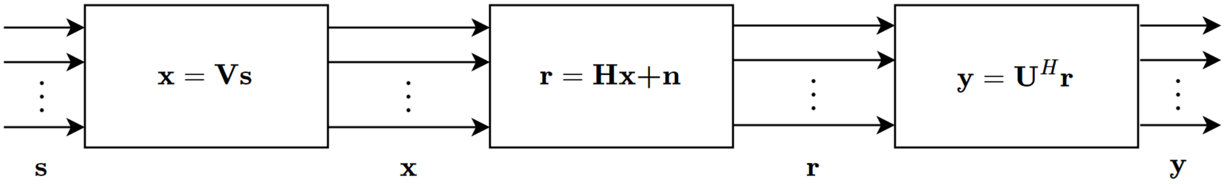

$\mathbf{x} = \mathbf{Vs}$
$\mathbf{r} = \mathbf{Hx}+\mathbf{n}$
$\mathbf{y} = \mathbf{U}^H\mathbf{r}$
s
x
r
y

Figure 2.12: MIMO Channel Decomposition with SVD [3].

The decomposition can be further illustrated through the simplification of the output signal vector $\mathbf{y}$ as follows.

$$\mathbf{y} = \mathbf{U}^H\mathbf{r} = \mathbf{U}^H(\mathbf{Hx} + \mathbf{n})$$

$$= \mathbf{U}^H(\mathbf{H}(\mathbf{Vs}) + \mathbf{n}) = \mathbf{U}^H(\mathbf{U}\boldsymbol{\Sigma}\mathbf{V^H}(\mathbf{Vs}) + \mathbf{n})$$

$$= \boldsymbol{\Sigma}\mathbf{s} + \mathbf{U^H n} = \boldsymbol{\Sigma}\mathbf{s} + \tilde{\mathbf{n}}$$

2.3 Software-Defined Radio

The ideal Software Radio (SR) is described in [17] as a communication system that implements all communication functions from antenna to speaker in software with Digital

Signal Processing (DSP). In this ideal system, parameters such as bandwidth, modulation, and frequency can be reconfigured by loading new software.

There are certain aspects of the transceiver which cannot be practically implemented in software such as analog filters for anti-aliasing, power amplifiers, Analog to Digital converters and Digital to Analog converters. These hardware components impose limitations on the bandwidth and frequency range achievable by the radio. Radios that use software to define a portion of the waveform are termed Software-Defined Radios (SDRs) [18]. SDRs possess much more flexibility than traditional radios which can be leveraged for a variety of applications.

Cognitive Radio (CR) is a one example of leveraging the flexibility of SDRs. CRs use input from their environment to alter the behavior of the radio accordingly. One use-case of a CR is to dynamically access spectrum as a secondary user while minimizing interference with the primary user of the channel. The radio performs spectrum sensing prior to transmission in order to ensure that the primary user is not occupying the channel.

An Universal Software Radio Peripheral (USRP) is a family of SDRs manufactured by Ettus Research which will be used extensively through this thesis. Key features of the USRP compared to other commercial SDRs are that the devices are relatively inexpensive and are Open-Source Hardware. USRPs have been largely adopted by open-source radio software like GNU Radio which makes it a useful tool for sharing research and promoting the replication of experimental results.

Chapter 3

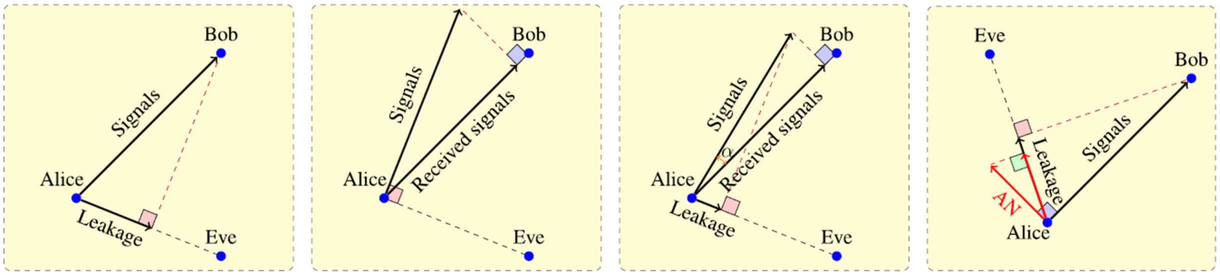
Physical Layer Security

Physical Layer Security (PLS) is a classification of security methods that take advantage of physical properties in the waveform or channel to secure communication. Over the past decade, advancements in Multiple-Input Multiple-Output (MIMO) systems have expanded the potential capabilities of PLS, meanwhile the development of technologies such as the IoT has provided new applications. While PLS has been heavily researched, literature that includes implementation is still developing. [19] analyzes the implementation of a phased-array beamformer masked with the Direction Modulation (DM) technique and [20] expands on this implementation with the development of a DM-enabled Digital Video Broadcast transmitter. [21] characterizes the performance of the Out-Phase Array Linearized Signaling technique developed in [22]. [23] describes the implementation of a system that performs secret key generation using shared channel characteristics tested on both Long-Term Evolution (LTE) and WiFi signals. The design work covered in this thesis attempts to add to this area of developing research by creating an open-source implementation of two PLS techniques that can be used directly with common SDR front-end devices to enable easy OTA experimentation and adaptation into new waveforms.

The focus of the last decade's MIMO PLS research has centered around the areas of secure multiantenna techniques and key generation. This chapter highlights the major advancements made in area of secure multiantenna techniques and covers the metrics used to evaluate PLS techniques in the literature.

3.1 Secure Multiantenna Techniques

PLS can be achieved through effective precoding at the transmitter. The examined techniques create a secrecy rate by attempting to degrade Eve's channel relative to Bob's. Four techniques that are representative of this area are Beamforming, Zero-Forcing (ZF), Convex Optimization (CVX), and Artificial Noise (AN). Figure 3.1, found in [4], describes each technique in terms of the transmission's orthogonality to Bob and Eve.



(a) Beamforming. (b) ZF precoding. (c) CVX-based precoding. (d) AN precoding.

Figure 3.1: Secure Multiantenna Techniques [4].

Beamforming is analyzed in the context of PLS as providing the best possible channel to Bob while indiscriminately allowing leakage to Eve. While Bob's capacity will be the highest for this technique, it does not necessarily provide the largest capacity difference between Bob and Eve.

ZF is precoding that places all transmissions completely orthogonal to Eve, thus forcing her capacity to zero. While Eve's capacity is the lowest with this technique, it does not account for how little information is received by Bob as a result. Therefore, the capacity difference between Bob and Eve isn't maximized in this case either.

CVX is the only technique that achieves secrecy capacity by maximizing the capacity difference between Bob and Eve. This is done by optimizing between beamforming towards Bob and ZF towards Eve. CVX is more computationally expensive than Beamforming or ZF.

While ZF and CVX provide secrecy benefits, they both rely on knowledge of Eve's channel in order to work. In the case of a passive eavesdropper where Eve's channel is not available, the AN technique can provide more secrecy than beamforming by injecting noise orthogonal to Bob on top of a beamformed information signal.

3.1.1 Generalized Singular Value Decomposition Beamforming

Generalized Singular Value Decomposition (GSVD) Beamforming takes advantage of Bob's and Eve's CSI to align the source information to the nullspace of Eve while optimally directing the transmission in Bob's subspace. The precoding applied is $\mathbf{x} = \mathbf{v}\mathbf{s}$ where the elements of s are complex Gaussian distributed RVs and $\mathbf{v}$ corresponds to the largest generalized eigenmode of $(\mathbf{I} + \mathbf{P}\mathbf{h}_\mathbf{b}^\mathbf{H}\mathbf{h}_\mathbf{b}, \mathbf{I} + \mathbf{P}\mathbf{H}_\mathbf{e}^\mathbf{H}\mathbf{H}_\mathbf{e})$, but this will be sub-optimal across all values of Signal-to-Noise Ratio (SNR) [24]. In the high SNR regime, the optimal beamforming approach is to direct the message to Bob while being constrained to remain in the nullspace of Eve. For low SNR, directing the message towards Bob without regard for Eve's nullspace is optimal. With only statistical CSI at Eve, a SVD scheme is proposed in [24] that can be adjusted close to Eve's nullspace and in Bob's subspace.

[12] presents a GSVD precoding scheme for MIMO that focuses on separating the spatial channels of the transmission into two different subspaces, S1 and S2. S1 is designed to be only seen by Bob while S2 can be seen by both Bob and Eve. The majority of the transmit power is transmitted into S1, and the power that is distributed to each subspace is split uniformly across the dimensions of that subspace [25]. At high SNR, this precoding achieves the secrecy capacity of the channel. At low SNR, this scheme does not achieve capacity. [26] describes another GSVD precoding scheme where the only spatial direc-

tions that are used are the ones where Bob sees a stronger signal than Eve. This scheme performs better than the subspace separation method at lower SNR.

3.1.2 Zero Forcing

[27] developed a Zero Forcing (ZF) precoding scheme that separates messages into different beamforming directions which are repeated at several antennas. The ZF precoding matrix F_z cancels out the gain of the wiretap channel and is designed such that there is no interference between antennas. The received signals of Bob and Eve are modeled as

$$\begin{bmatrix} \mathbf{y_b} \\ \mathbf{y_e} \end{bmatrix} = \begin{bmatrix} \mathbf{H_b} \\ \mathbf{H_e} \end{bmatrix} \mathbf{F_z x}$$

Where the security constraint is $\mathbf{H_e F_Z} = \mathbf{0}$ and the reliability constraint is $\mathbf{F_Z^{-1}} \begin{bmatrix} \mathbf{H_b} \\ \mathbf{H_e} \end{bmatrix} \mathbf{F_Z} = \mathbf{I}$.

This scheme requires Alice to have more antennas than Eve. ZF will provide a larger secrecy rate than GSVD when the number of antennas at Alice, Bob and Eve satisfy $N_b + N_e \leq N_a$.

This scheme is of particular interest, because it introduces the idea of using Mean Squared Error (MSE) as a metric for secrecy instead of equivocation. MSE will not guarantee perfect information theoretic security, rather it only provides a finite level of confusion for the eavesdropper. This is framed in the context of applications that require a minimum MSE to function. An example given is a secure video broadcasting service where the video provider wishes to provide subscribed users a high quality video while limiting unsubscribed users to degraded video.

Another way to look at this scheme is that filters are designed at Alice and Bob such that the MSE of Bob is kept above a threshold (the reliability constraint) while keeping the

MSE of Eve below a target value (the secrecy constraint).

The authors of [27] claim that their analysis addresses PLS is "from the estimation-theoretic point of view, rather than the information-theoretic viewpoint."

3.1.3 CVX

CVX is a convex optimization solver that can be used with ZF and AN schemes to precode with minimal CSI. Methods in the literature use an exhaustive search to find an optimal transmit covariance matrix that can maximize the secrecy rate of the scheme. CVX solvers are written in MATLAB, allowing constraints and objectives to be incorporated into MATLAB models.

3.1.4 Artificial Noise Generation

[28] proposed an Artificial Noise (AN) Generation scheme where Alice divides her power between transmitting a message to Bob in his subspace and transmitting Gaussian noise into Bob's nullspace. Assuming Bob's and Eve's channels are independently faded, Eve will see some of the AN in her subspace. This technique's major strength is that the secrecy provided scales well with SNR since an increase in SNR at Eve will increase the received AN power as well as the message power. Another interesting property of this scheme is that the communication's secrecy does not depend on the secrecy of Bob's CSI [28]. This means that Bob can transmit his CSI gains directly to Alice without fear of them being intercepted by Eve.

The AN technique is a focus of the implementation work in this thesis and will be expanded on in chapter 4.

3.2 Metrics for Comparing PLS Techniques

There are a vast array of different strategies for achieving confidentiality at the physical layer. Each technique requires certain assumptions to be made about the communication system's resources and knowledge. To understand how these techniques compare to each other, it is useful to define the assumptions that can be made about the system as well as the different metrics used to evaluate the system's security.

3.2.1 Channel State Information

CSI is one of the most fundamental assumptions made in PLS. CSI refers to each communicant's knowledge of their own channel, and the channels between other users. Figure 3.2 depicts a typical MIMOME channel where the channel matrices **H** and **G** contains the gains for the Alice-Bob and Alice-Eve channels, respectively.

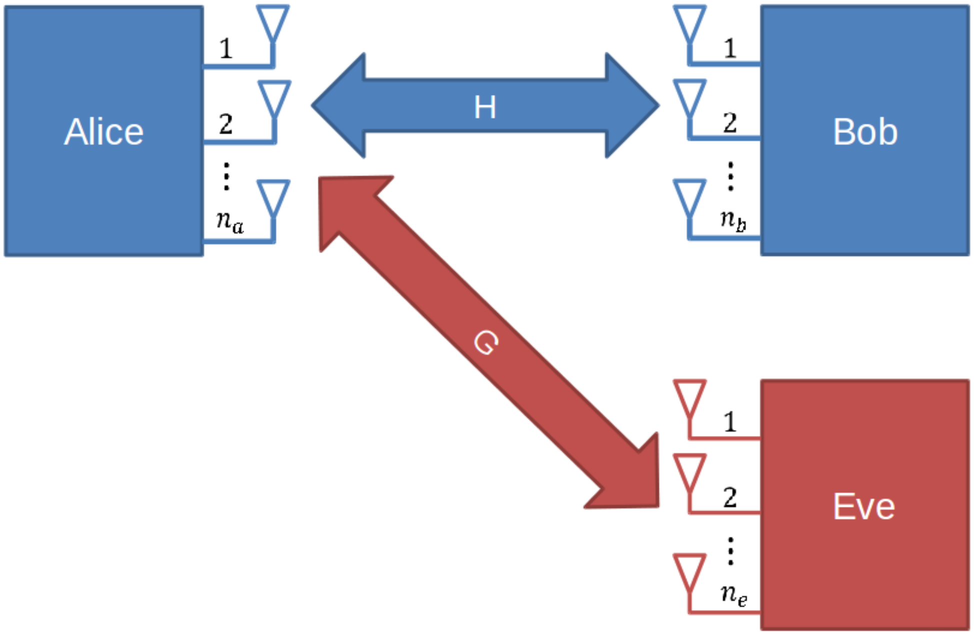


Figure 3.2: MIMOME Channel with Alice-Bob Channel **H** and Alice-Eve Channel **G**.

Bob and Eve can learn their respective channels through a number of channel estimation techniques. Channel estimation is relatively easy to accomplish through data-aided schemes involving unique words or through dedicated sub-carriers in the case of Orthogonal Frequency Domain Multiplexing (OFDM).

Things start to become more complicated when Alice needs to know her channel to Bob. Bob can either sound the channel for Alice or directly send Alice the channel gains. From a security standpoint, if Bob's channel gains are sent OTA to Alice, Eve is assumed to intercept them and therefore know both channels.

By performing her own channel estimate, Alice can prevent Eve from knowing Bob's channel. This is beneficial from a security perspective, but comes at a few costs. First, we must assume reciprocity between the transmitter and receiver in order to apply this method. Channel reciprocity can't be assumed in all cases, such as Frequency-Division Duplex (FDD) systems where the uplink and downlink are on different frequencies. The RF chains of Alice and Bob are also not generally assumed to be reciprocal and must be calibrated to assume reciprocity [29]. Second, performing another channel estimation at Alice increases her complexity. Lastly, a second round of estimation error is incurred when Alice estimates the channel gains independently from Bob. Therefore, unless the secrecy of Bob's CSI is necessary for the secrecy of communication, it is usually preferable for Bob to send his CSI explicitly to Alice.

CSI at the Transmitter (CSIT) and CSI at the Receiver (CSIR) can be categorized as: full instantaneous CSI, deterministic imperfect CSI, indeterministic imperfect CSI, or statistical CSI [5]. A realistic assumption in the case of a passive eavesdropper would be that Alice has deterministic imperfect CSI for Bob, and statistical CSI for Eve. Many of the precoding techniques that were discussed assume full or imperfect knowledge of Eve's CSI which is unrealistic in the case of a passive eavesdropper.

3.2.2 Secrecy Measures

To compare PLS techniques, [4] uses two metrics: secret channel capacity and computational complexity. Secret channel capacity, commonly referred to as secrecy capacity, is a measure of the Low Probability of Intercept (LPI) characteristics of a technique that provide information-theoretic security.

The One-Time Pad (OTP) is a theoretically unbreakable encryption technique first described by Frank Miller in 1882 to secure telegraph communications [30].

Plaintext is XORed with a completely random (non-repeating) OTP which serves as a key and is communicated in a completely secure manner. Shannon later proved that the OTP was information-theoretic secure. This means that given the cipher text, a reader could not learn anything about the plain text except the maximum possible length [31]. Regardless of the computational power of an adversary, the OTP could not be broken.

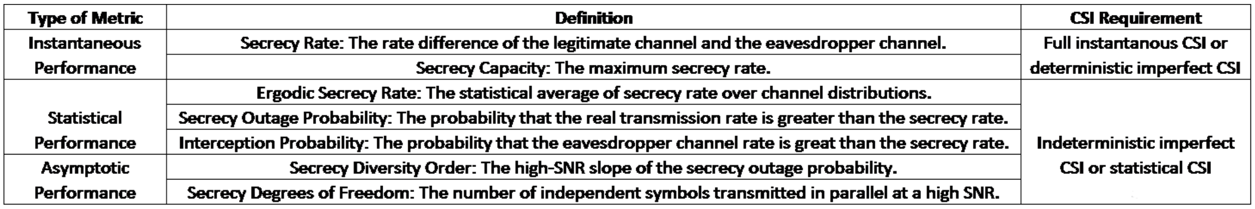
Wyner showed that for discrete memoryless channels where the eavesdropper's channel is degraded, information-theoretic secure communication could take place at rates below the secrecy capacity of the channel defined as the difference in the capacity of the intended receiver and the eavesdropper [1].

$$C_s = C_B - C_E$$

Leung later generalized this result to Gaussian channels [32].

Generally, an eavesdropper of encrypted communications requires a secret key to successfully receive the transmitted message. When this is the case, the amount of possibilities for the secret key will be directly proportional to the security of the encryption. This is when the eavesdropper's only option is to perform an exhaustive search over all possible keys. This metric is applicable to a few PLS schemes that rely on sequences or codebooks for security.

[5] produced Figure 3.3 to illustrate how the availability of CSI affects the performance metrics available to measure security in PLS.



Type of Metric	Definition	CSI Requirement
Instantaneous Performance	Secrecy Rate: The rate difference of the legitimate channel and the eavesdropper channel.	Full instantanous CSI or deterministic imperfect CSI
	Secrecy Capacity: The maximum secrecy rate.	
Statistical Performance	Ergodic Secrecy Rate: The statistical average of secrecy rate over channel distributions.	Indeterministic imperfect CSI or statistical CSI
	Secrecy Outage Probability: The probability that the real transmission rate is greater than the secrecy rate.	
	Interception Probability: The probability that the eavesdropper channel rate is great than the secrecy rate.	
Asymptotic Performance	Secrecy Diversity Order: The high-SNR slope of the secrecy outage probability.	
	Secrecy Degrees of Freedom: The number of independent symbols transmitted in parallel at a high SNR.	

Figure 3.3: Performance Metrics in Physical Layer Security[5].

Note that these CSI assumptions are required at both the intended receiver and the eavesdropper. For a passive eavesdropper, Alice will know at most the statistical CSI, therefore, secrecy rate and secrecy capacity will not be achievable metrics.

Chapter 4

Theory and Simulation Work

This chapter introduces the two PLS techniques implemented in this thesis, AN and PEAC, are introduced.

To verify and further understand the behavior of the PEAC and AN systems, simulations of both techniques are conducted in MATLAB.

4.1 Artificial Noise Generation

4.1.1 System Model

[28] proposes an AN precoding scheme where Alice divides her power between transmitting a message to Bob and transmitting Gaussian noise into Bob's nullspace. Assuming Bob's and Eve's channels are independently faded, Eve will see some of the AN in her rangespace. This technique's major strength is that the secrecy provided scales well with SNR since an increase in SNR at Eve will increase the received AN power along with the message power.

To construct the AN, Alice must know Bob's CSI. For a Rayleigh channel with flat fading, Bob only needs to either sound the channel or relay his channel coefficients to Alice faster than the coherence time of the channel. Another interesting property of this scheme is that the communication's secrecy does not depend on the secrecy of Bob's CSI [28]. This means that Bob can transmit his CSI directly to Alice without fear of it being intercepted by Eve.

Alice transmits

$$\mathbf{x}_k = \mathbf{s}_k + \mathbf{w}_k$$

where $\mathbf{x}_k$ and $\mathbf{w}_k$ are complex Gaussian vectors and $\mathbf{w}_k$ is chosen to lie in the nullspace of $\mathbf{h}_k$ by satisfying

$$\mathbf{H}_k\mathbf{w}_k = 0.$$

The AN term, $\mathbf{w}_k$, is generated from

$$\mathbf{w}_k = \mathbf{Z}_k\mathbf{v}_k$$

where $\mathbf{Z_k}$ is a unitary matrix that is the orthonormal basis for the nullspace of $\mathbf{H_k}$. Since Eve may be in a channel realization that aligns her nullspace with Bob's, the best strategy is to make each element of $\mathbf{v_k}$ a Gaussian distributed random variable. By doing this, the AN is generated randomly from the available orthonormal basis vectors for Bob's nullspace. The number of possible basis vectors for Bob's nullspace, N_{null}, is limited by the difference in the array sizes of Alice and Bob

$$N_{null} = N_{Alice} - N_{Bob} \quad \text{for} \quad N_{Alice} \geq N_{Bob}.$$

The signal received by Bob is

$$\mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{n}_k$$

$$\mathbf{z}_k = \mathbf{H}_k(\mathbf{s}_k + \mathbf{w}_k) + \mathbf{n}_k$$

$$\mathbf{z}_k = \mathbf{H}_k\mathbf{s}_k + \mathbf{n}_k.$$

The signal received by Eve is

$$\mathbf{y}_k = \mathbf{G}_k\mathbf{x}_k + \mathbf{e}_k$$

$$\mathbf{y}_k = \mathbf{G}_k(\mathbf{s}_k + \mathbf{w}_k) + \mathbf{e}_k$$

$$\mathbf{y}_k = \mathbf{G}_k\mathbf{s}_k + \mathbf{G}_k\mathbf{w}_k + \mathbf{e}_k$$

where the $\mathbf{G}_k\mathbf{w}_k$ represents the additional noise seen by Eve.

Figure 4.1 displays an example of a wiretap code being applied to the AN scheme. To encode data with the wiretap code, one of four possible constellation points (one in each quadrant) are chosen for each of the four symbols. This scenario is the same wiretap code introduced in section 2.1.3, where Bob can only see the source information and has a SNR large enough to demodulate 16-QAM. Eve sees AN along with the source information and only has a SNR large enough to demodulate QPSK. When Eve receives a transmitted symbol, she can only tell which quadrant it is in and since all of the source symbols can map to points in every quadrant, it is completely ambiguous to Eve which symbol was sent. The wiretap coding adds redundancy by mapping a single symbol to four possible constellation points and therefore the effective transmission rate is reduced by half which agrees with the theoretical calculation of the secrecy rate

$$R_S = R_B - R_E$$

$$R_S = 4\,\text{bits/sym} - 2\,\text{bits/sym} = \textbf{2 bits/sym}.$$

It is important to emphasize that this scenario will only exist when Eve's receiver is in an edge case where it can do absolutely no better than QPSK.

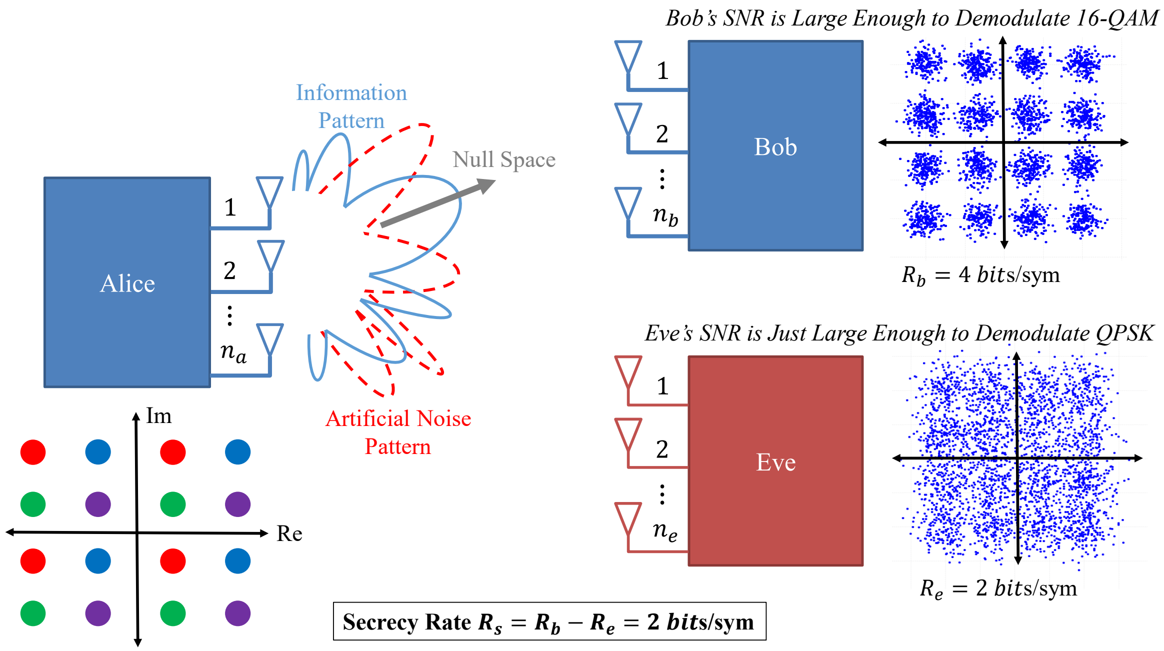

Bob's SNR is Large Enough to Demodulate 16-QAM
Information Pattern
Null Space
Alice
1
2
⋮
n_a
Bob
n_b
$R_b = 4$ bits/sym
Eve's SNR is Just Large Enough to Demodulate QPSK
Eve
n_e
$R_e = 2$ bits/sym
Artificial Noise Pattern
Im
Re
Secrecy Rate $R_s = R_b - R_e = 2$ bits/sym

Figure 4.1: An example wiretap coding scheme applied to the Artificial Noise technique.

4.1.2 AN Simulations

Simulation Description

The AN technique is simulated in Matlab as a masked beamformer described in [24] where Alice has 2 antennas while Bob and Eve both have a single antenna. Alice uses the transmit beamforming technique shown in Figure 4.2.

The transmitter sends signals

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \beta_1 s \\ \beta_2 s \end{bmatrix}.$$

The received signal is

$$r = \mathbf{hx} + n$$

$$r = h_1 x_1 + h_2 x_2 + n$$

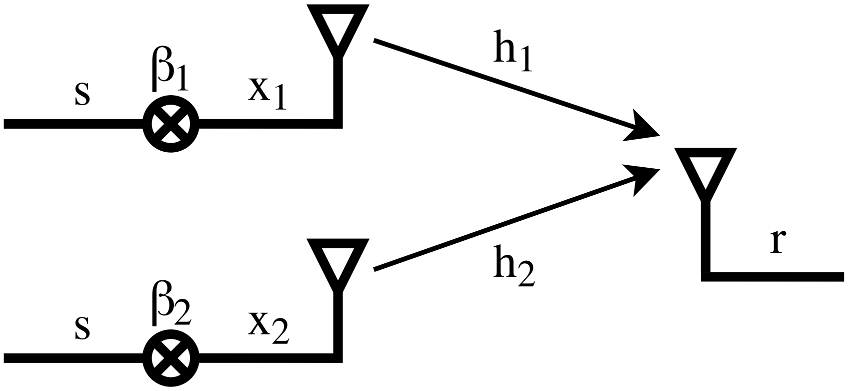


Figure 4.2: 2×1 Transmit Beamformer.

$$r = h_1 \beta_1 s + h_2 \beta_2 s + n.$$

Beamforming weights are chosen to be normalized phase shifts

$$\beta_k = \frac{h_k^*}{|h_k|}, \quad \text{for} \quad k = 1, ..., M_t$$

where M_t is the number of transmit antennas [33]. The received signal becomes

$$r = \frac{h_1 h_1^* s}{|h_1|} + \frac{h_2 h_2^* s}{|h_2|} + n = (|h_1| + |h_2|)s + n.$$

Additionally, the simulation uses QPSK modulation and a Rayleigh flat-faded channel.

The power tradeoff between the information signal and AN signal is parameterized with α, the ratio of AN amplitude to total amplitude. The transmitted symbol, x, with total power P is

$$\sqrt{P} x_k = \sqrt{(1-\alpha)P} s + \sqrt{\alpha P} w.$$

The AN simulation has a unity transmit power constraint and the resulting transmit signal is

$$x_k = \sqrt{1-\alpha} s + \sqrt{\alpha} w.$$

Simulation Results

Figure 4.3 shows the simulation results for the case where there is no AN added. For comparison, the theoretical BER curves for a 1×2 Single-Input Multiple-Output (SIMO) system using Maximal Ratio Combining (MRC) and a 1×1 Single-Input Single-Output (SISO) system in a Rayleigh channel are also plotted along side the simulation results.

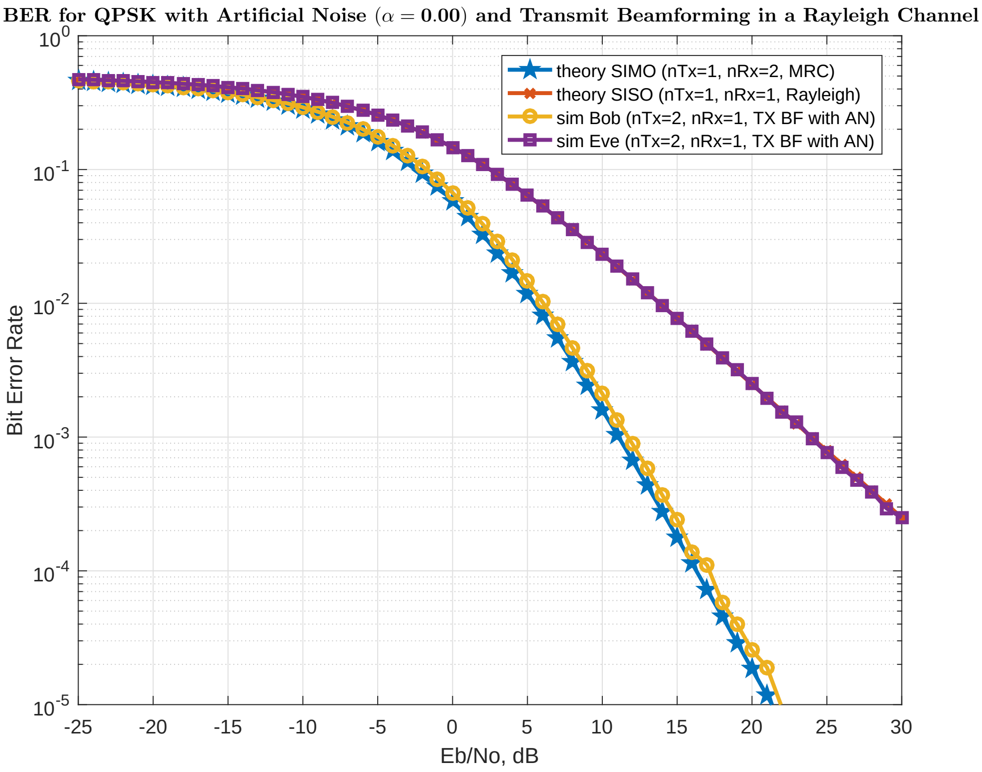


Figure 4.3: Simulated BER curves for Bob and Eve with TX Beamforming and no AN added.

Bob's BER curve in Figure 4.3 agrees with the theoretical performance of the MRC algorithm. The MRC algorithm will slightly outperform TX beamforming because the noise scaling provided by the two algorithms slightly favors MRC. Eve's BER curve is experiencing the same performance as a SISO system in a Rayleigh channel because Eve will not receive the diversity gain that Bob gets from Alice's TX beamforming.

Figures 4.4, 4.5, and 4.6 show the simulation results of the AN system with a power ratios of $\alpha = 0.25$, $\alpha = 0.50$, and $\alpha = 0.75$, respectively.

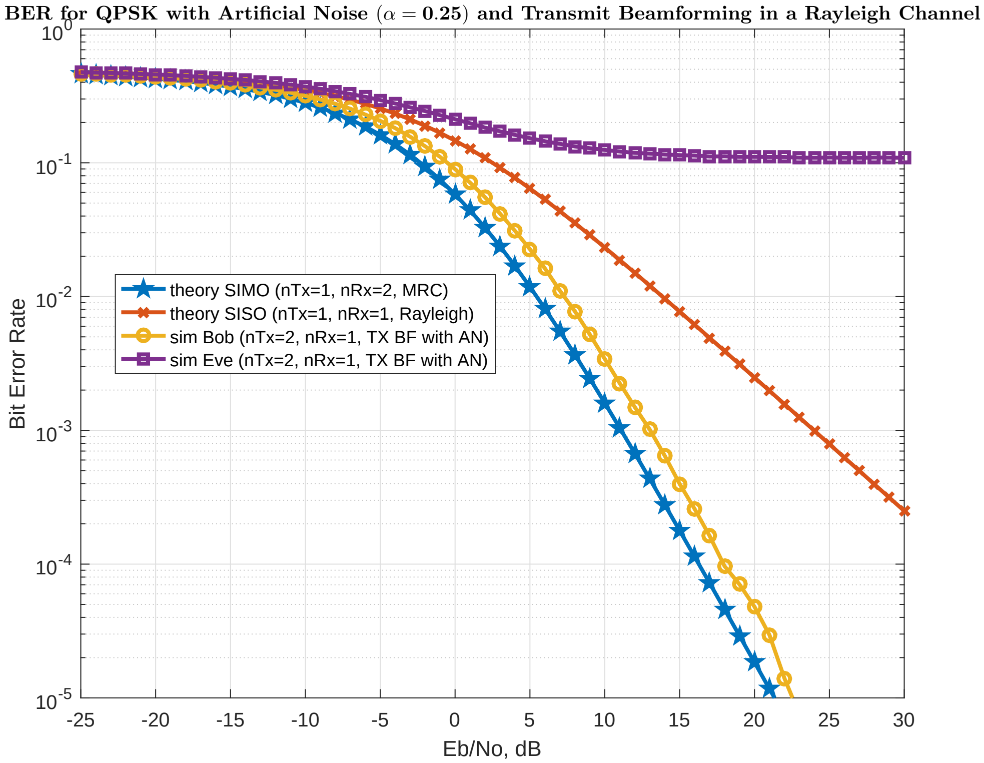


Figure 4.4: Simulated BER curves for Bob and Eve with TX Beamforming and AN added with a power ratio of $\alpha = 0.25$.

Figure 4.4 shows the effects of devoting a quarter of the total transmission power to AN. Eve's BER is drastically increased at higher SNRs and even appears to be asymptotically approaching a limit in the high SNR regime. This limit makes sense conceptually since an increase in Eve's received SNR will increase the SNR of the AN along with the information. For low SNR, there will be a decrease in BER from 0.5 until the SNR of information signal reaches the Signal-to-Interference Ratio (SIR) caused by the AN, which forms the high SNR regime limit. Therefore, Eve's system can outperform Bob's in terms of received SNR and still benefit very little or not at all from it! In comparison to Bob's BER curve from Figure 4.3, the performance gain of Bob's curve in 4.4 has diminished. This is due to

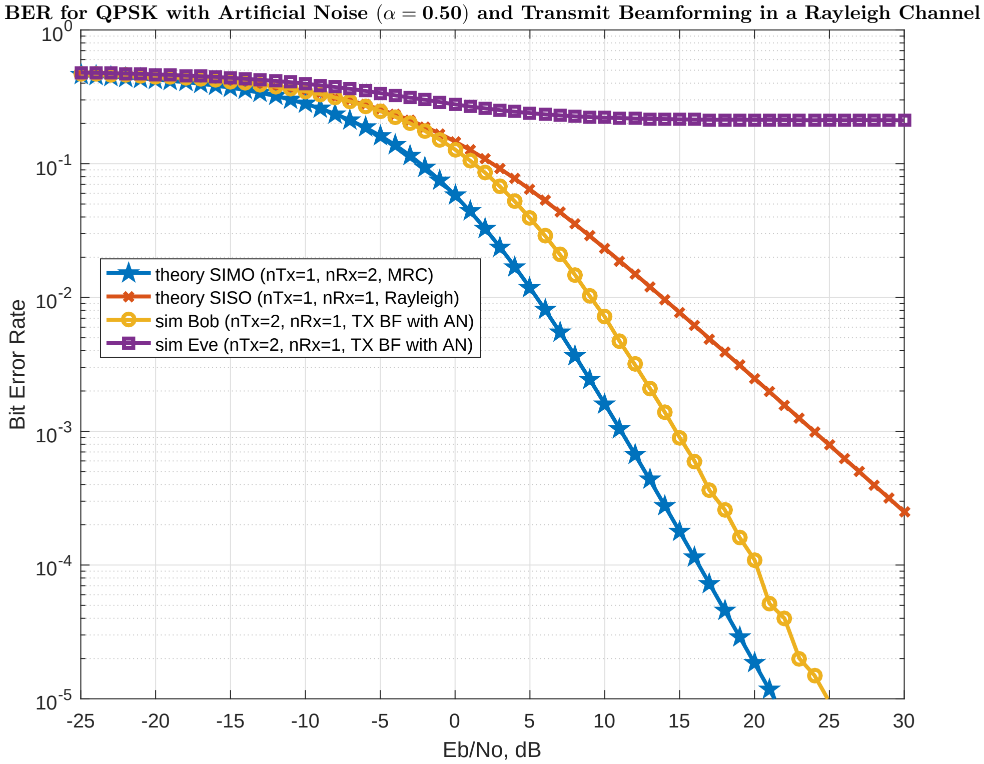


Figure 4.5: Simulated BER curves for Bob and Eve with TX Beamforming and AN added with a power ratio of $\alpha = 0.50$.

the fact that 25% of the transmit power that was originally going towards the information signal is now being used for the AN signal.

These trends can be further illustrated by examining Figures 4.5 and 4.6. As the power going towards the AN (α) increases, Eve's BER asymptote increases and Bob's performance gain decreases. The optimal amount of AN power used will depend on the BER of both Bob and Eve in a given scenario, but will tend to be around $\alpha = 0.5$ for high SNR cases [28].

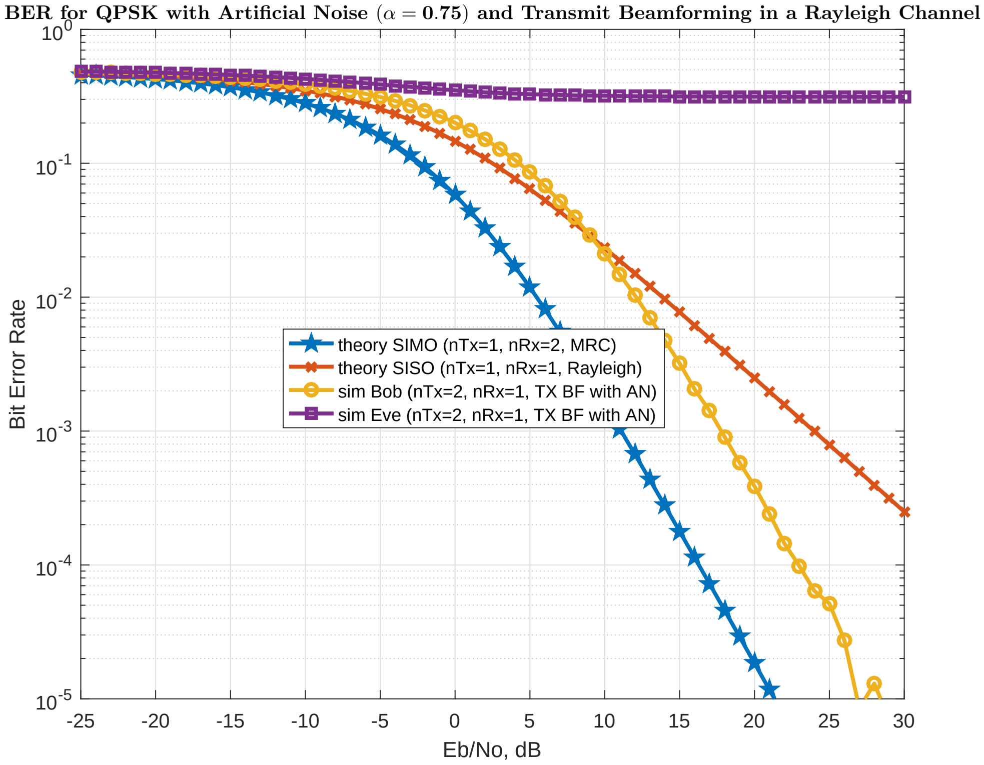


Figure 4.6: Simulated BER curves for Bob and Eve with TX Beamforming and AN added with a power ratio of $\alpha = 0.75$.

4.2 Phase-Enciphered Alamouti Coding

4.2.1 System Model

Alamouti Space-Time Block Code

In [34], Alamouti proposed a Space-Time Block Code (STBC) to achieve transmitter diversity. Prior to this, diversity techniques were only applied at the receiver using algorithms such as MRC. These techniques required multiple receiver antennas and were often impractical to implement on mobile handsets in cellular networks, since doing so would require extra RF chains for every phone on the network. As a result, diversity techniques were only used at base stations to improve their reception quality. Alamouti showed that it was possible to achieve the same advantages, or diversity order, with a technique requiring multiple transmit antennas and a single receive antenna. Furthermore, he showed that the transmit data can be separated at the receiver with only linear computational complexity, making the processing requirements comparable to MRC. An additional feature of the Alamouti coding scheme is that it provides diversity gains independent of the degeneracy of the channel.

The Alamouti coding scheme is usually described by the 2×2 matrix

$$\mathbf{C} = \begin{bmatrix} s_1 & -s_2^* \\ s_2 & s_1^* \end{bmatrix}$$

where the columns represent timeslots and the rows represent different transmit antennas.

Secure STBCs without Transmitter CSI

For many PLS techniques, Alice needs to have an accurate estimate of Bob's CSI. When Alice's CSI estimate is inaccurate, these techniques become less reliable and may even ill-condition the environment for Bob. For the AN scheme, the inaccuracy in Alice's knowledge of Bob's CSI directly corresponds to AN leakage into Bob's channel.

The author of [35] introduces a technique to achieve a secure STBC without needing to estimate CSI at the transmitter. This technique relies on a mutual RSSI measurement in order to seed a psuedo-random sequence used to secure communication. The psuedo-random sequence will determine phase shifts, θ_1 and θ_2, that are applied to each transmit element in the Alamouti STBC. Each phase shift is applied for one code duration. For a single codeword, the transmitter encodes source information, s_1 and s_2, as

$$\mathbf{X} = \begin{bmatrix} s_1 e^{j\theta_1} & s_2 e^{j\theta_2} \\ -s_2^* e^{j\theta_1} & s_1^* e^{j\theta_2} \end{bmatrix}$$

where the rows of $\mathbf{X}$ represent different transmit antennas and the columns correspond to time slots. Notice that this modified scheme still preserves the orthogonality of the Alamouti coding scheme. Figure 4.7 describes this scheme for a 2 × 1 system.

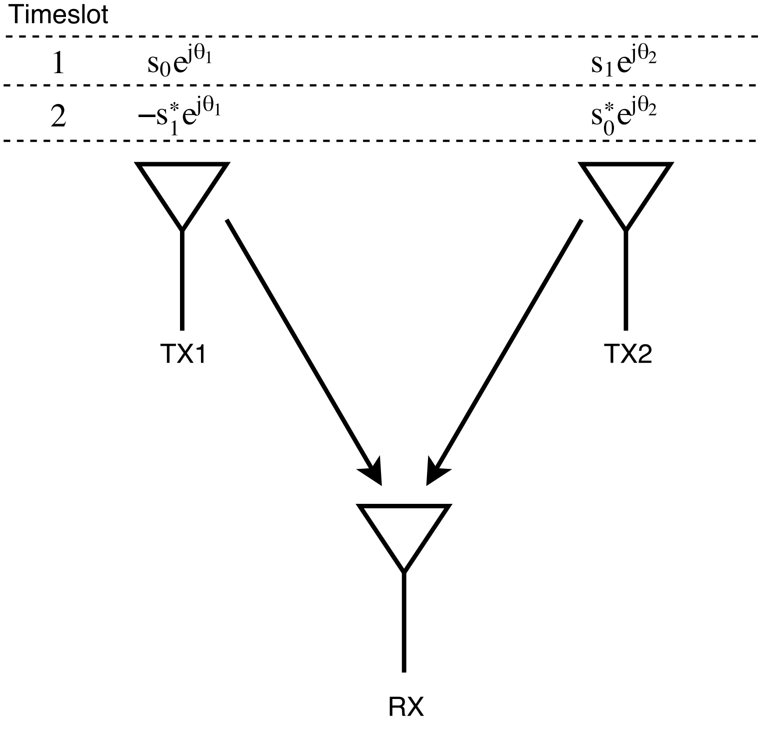


Figure 4.7: Phase-Enciphered Alamouti Coding Scheme.

This is an application of the key-generation branch of PLS, but the key is used to encrypt the transmission at the physical layer rather than using a higher layer protocol. In the implementation described in this thesis, the key generation is handled in software and the design is focused on the realization of the physical layer encryption.

Bob's received signal is

$$\mathbf{z} = \mathbf{X}\mathbf{h} + \mathbf{n}$$

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} s_1 e^{j\theta_1} & s_2 e^{j\theta_2} \\ -s_2^* e^{j\theta_1} & s_1^* e^{j\theta_2} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \end{bmatrix}$$

Which can be represented more conveniently as

$$\begin{bmatrix} z_1 \\ -z_2^* \end{bmatrix} = \begin{bmatrix} h_1 e^{j\theta_1} & h_2 e^{j\theta_2} \\ -h_2^* e^{j\theta_1} & h_1^* e^{j\theta_2} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} + \begin{bmatrix} n_1 \\ -n_2 * \end{bmatrix}$$

$$\tilde{\mathbf{z}} = \mathbf{H}(\theta_1, \theta_2)\mathbf{s} + \tilde{\mathbf{n}}$$

Using the MRC algorithm, the source information can be estimated as

$$\hat{\mathbf{s}} = \mathbf{H}^{+}(\theta_1, \theta_2)\tilde{\mathbf{z}}$$

where $\mathbf{H}^{+} = (\mathbf{H}^{*}\mathbf{H})^{-1}\mathbf{H}^{*}$ is the Moore-Penrose pseudoinverse of $\mathbf{H}$.

Eve's received signal is

$$\mathbf{y} = \mathbf{X}\mathbf{g} + \mathbf{e}$$

which can be similarly decomposed into

$$\tilde{\mathbf{y}} = \mathbf{G}(\theta_1, \theta_2)\mathbf{s} + \tilde{\mathbf{e}}$$

where the source information can also be solved for

$$\hat{\mathbf{s}} = \mathbf{G}^{+}(\theta_1, \theta_2)\tilde{\mathbf{y}}.$$

In order to calculate ŝ, Eve will need an accurate estimate of $\mathbf{G}(\theta_1, \theta_2)$. Eve is presumed to have perfect CSI, and therefore knowledge of g. Eve does not know the phase rotations, θ_1 and θ_2, but she will know the L^2 possible combinations of them. An exhaustive search over the phase rotations along with the L^2 possible symbol combinations will therefore require a search complexity of $O(L^4)$ using Big O notation. Even with the brute-force search, Eve still makes an ambiguous decisions between L equally valid combinations of phase shifts and sent symbols.

[35] generalizes Eve's maximum likelihood detector to discuss the relationship between the number of phase rotatations, N_{Rot}, applied at Alice and Eve's diversity order, D_{Eve}. This relationship can be characterized as

$$D_{Eve} = N_{Bob} - N_{Rot}.$$

This thesis will only focus on the case where Alice applies the maximum number of phase rotations to completely deny Eve access to the source information.

Additionally, [35] covers the design of a 4^{th} order STBC and generalizes this technique to securing STBCs of an arbitrary order. This thesis will examine the 2×1 Alamouti case.

4.2.2 PEAC Simulations

The PEAC simulation uses QPSK modulation and rotations for both transmit antennas. Bob uses a shared knowledge of the phase rotations to undo them during Alamouti equalization while Eve uses a matched filter bank in an attempt to isolate the transmitted symbol-phase rotation pair with a Minimum Mean Square Error (MMSE) estimator as described in [35]. Figure 4.8 shows the BER curves for Bob and Eve with PEAC applied. The theoretical BER curves for a SISO system in a Ralyeigh channel, Alamouti Coding, and MRC are also plotted for comparison.

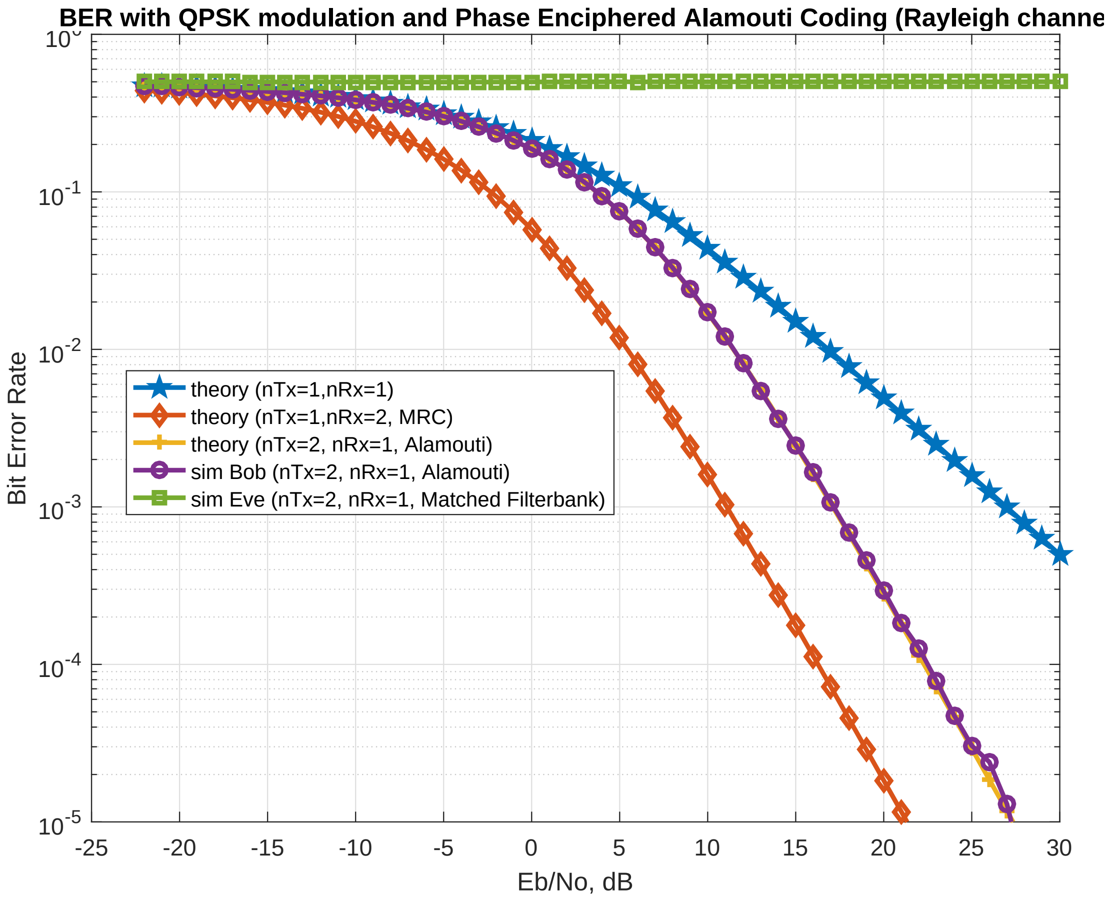


Figure 4.8: Simulated BER curves for Bob and Eve with Alice applying PEAC.

As expected, Eve's BER is indicative of her being unable to decode any of the transmitted information, because she is making ambiguous guesses at the transmitted symbol-phase shift pairs. Bob's performance gain agrees with the theoretical curve for Alamouti coding and performs worse than MRC, but better than a SISO system.

Chapter 5

Experimental Setup

The design of both the PEAC and AN systems were done using GNU Radio, an open source software toolkit designed for real-time signal processing on a SDR. The PEAC and AN systems were then both evaluated in a testbed of USRP SDRs.

This chapter begins with an introduction of GNU Radio. Next, the requirements for constructing an OTA communications system in GNU Radio are introduced using a single-carrier QPSK waveform as an example. Finally, the testbed used for experimentation is described.

5.1 GNU Radio Introduction

GNU Radio possesses a library of pre-developed DSP functions in it's core framework which are implemented in blocks. Each block usually completes an independent function such as low-pass filtering or automatic gain control. Blocks are written in either C++ or Python and connected together in a Python flowgraph. The flowgraph interfaces with the GNU Radio scheduler to manage streams of samples between blocks. Additionally, flowgraphs will define variables that are in the scope of all blocks inside that flowgraph, allowing something like sample rate to be set common to many blocks. GNU Radio Companion (GRC) is a graphical interface for constructing GNU Radio flowgraphs from processing blocks.

5.1.1 Blocks

Blocks allow GNU Radio to implement DSP in modular stages. A block can have any number of input and output ports which are used to interface with other blocks. Ports are defined by their data type. Table 5.1 shows some common port data types in GNU Radio.

Table 5.1: Common port types in GNU Radio [8].

Data Type	**GRC Color Mapping**	**Description**
Complex	Blue	2 x 32-bit Floating Point
Float	Orange	32-bit Floating Point
Int	Green	32-bit Signed Integer
Short	Yellow	16-bit Signed Integer
Char	Pink	8-bit Signed Integer
Async Message	Gray	Asynchronous Polymorphic Type

There are several block types which are defined by port locations and the relative data rates of those ports. Table 5.2 shows various block types in GNU Radio.

5.1.2 Flowgraphs

While each block implements an individual process, signal processing applications will almost always require multiple block operations to be carried out sequentially and/or in parallel. GNU Radio implements this flow of data items between blocks in a flowgraph. Flowgraphs manage the connections between blocks, defining block parameters,

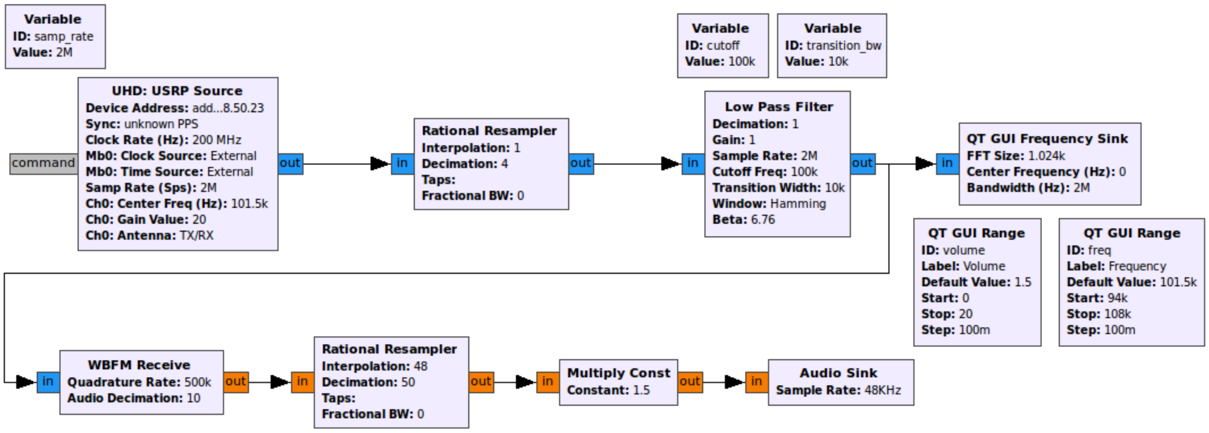
Table 5.2: Block types in GNU Radio [8].

Block Type	Description	Example Block
Synchronous	The number of input items equals the number of output items (1:1)	Multiply Const
Decimation	The number of input items is a fixed multiple of the number of output items (N:1)	Low Pass Filter
Interpolation	The number of output items is a fixed multiple of the number of input items (1:M)	Interpolating FIR Filter
General/Basic	No relation between the number of input and output items (N:M)	Rational Resampler
Sink	Only input ports	Audio Sink
Source	Only output ports	UHD: USRP Source

and interfacing with the Graphical User Interface (GUI) that controls the SDR while it is running.

GRC is the GUI that can be used by the end-user to design flowgraphs. Each GRC file represents a single flowgraph, consisting of blocks and global variables. A flowgraph generated with GRC is written in Python, but flowgraphs can also be manually written in C++. Figure 5.1 shows an example flowgraph used to demodulate Wide-Band Frequency Modulated (WBFM) signals such as radio stations.

Figure 5.1 shows examples of several block types and data types. Notice that data types must agree between connections and that this flowgraph contains a type of block not defined in Table 5.2 - blocks with no ports. These blocks can define variables and objects in the scope of the flowgraph or interface directly with the GUI, but will not directly operate on items.



Variable
ID: samp_rate
Value: 2M

UHD: USRP Source
Device Address: add...8.50.23
Sync: unknown PPS
Clock Rate (Hz): 200 MHz
Mb0: Clock Source: External
Mb0: Time Source: External
Samp Rate (Sps): 2M
Ch0: Center Freq (Hz): 101.5k
Ch0: Gain Value: 20
Ch0: Antenna: TX/RX

Rational Resampler
Interpolation: 1
Decimation: 4
Taps:
Fractional BW: 0

Variable
ID: cutoff
Value: 100k

Variable
ID: transition_bw
Value: 10k

Low Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 2M
Cutoff Freq: 100k
Transition Width: 10k
Window: Hamming
Beta: 6.76

QT GUI Frequency Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 2M

QT GUI Range
ID: volume
Label: Volume
Default Value: 1.5
Start: 0
Stop: 20
Step: 100m

QT GUI Range
ID: freq
Label: Frequency
Default Value: 101.5k
Start: 94k
Stop: 108k
Step: 100m

WBFM Receive
Quadrature Rate: 500k
Audio Decimation: 10

Rational Resampler
Interpolation: 48
Decimation: 50
Taps:
Fractional BW: 0

Multiply Const
Constant: 1.5

Audio Sink
Sample Rate: 48KHz

Figure 5.1: GRC Flowgraph for a FM Receiver [6].

5.1.3 Customization

GNU Radio possesses a large library of pre-developed DSP functions in it's core framework, but users may still need to design customized blocks for their applications. Custom blocks can be written in 3 different programming languages depending on the processing speed required. Generally, a faster implementation is more difficult to program. Table 5.3 compares the different programming options for GNU Radio blocks.

Table 5.3: Customization options in GNU Radio.

Implementation	**Method**	**Speed**	**Difficulty**
Python	Embedded Python Block or OOT Module	Slow	Easy
C++	OOT Module	Medium	Medium
FPGA	RFNoC	Fast	Hard

Embedded Python blocks are one of the easiest way to produce a custom block. The Python block is included in the core block library and can be accessed directly from GRC. By opening an editor in the block's properties, unique functionality can be written directly without having to worry about the block's integration into the programming environment.

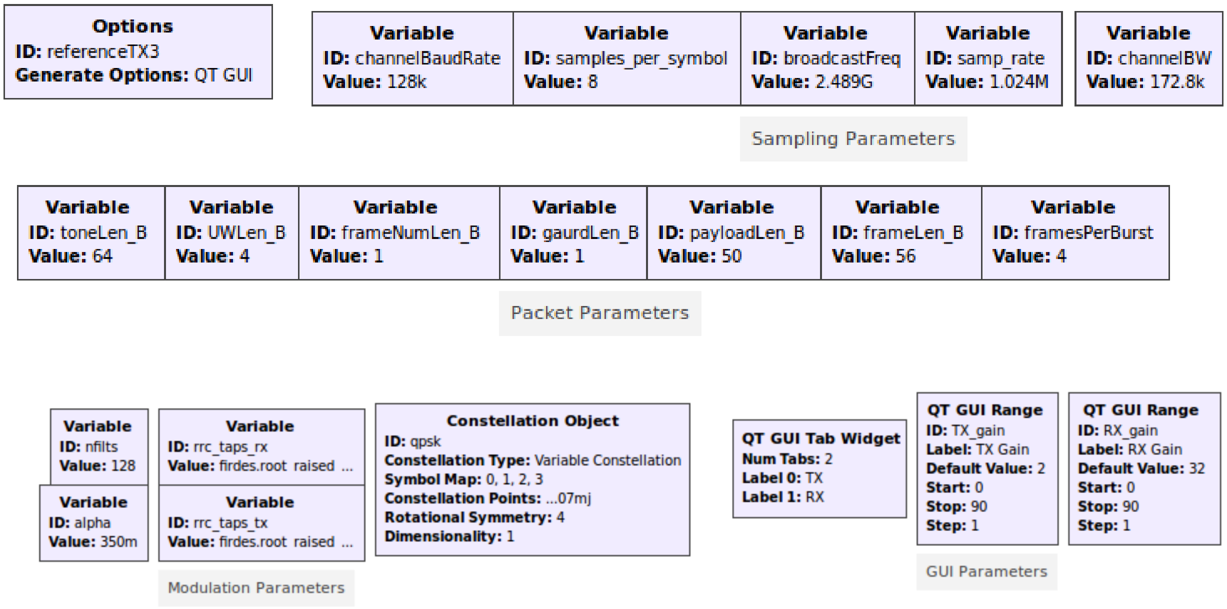
Out of Tree (OOT) Modules are modules existing separate from the GNU Radio source tree. This allows the user to maintain their code independent to the core GNU Radio framework [36]. Creating OOTs has been made much easier with gr_modtool, a script that edits makefiles and generates coding templates. OOTs can be used to write blocks in either Python or C++.

Radio Frequency Network on Chip (RFNoC) is a network-distributed heterogeneous processing tool that is designed to enable FPGA processing on USRP devices. RFNoC has made FPGA programming much easier than traditional methods, but it is still more challenging than writing an OOT module.

Additionally, Hierarchical blocks can be created in GRC to reduce the size of a complicated flowgraph by breaking it into parts. GRC will currently generates Hierarchical blocks in python, but they can also be written in C++ like flowgraphs.

5.2 QPSK Transmitter

The GRC flowgraph shown in Figure 5.3 depicts the transmitter designed for an OTA QPSK transmission of a single-carrier waveform. Figure 5.2 shows the parameter blocks for this flowgraph.



Options
ID: referenceTX3
Generate Options: QT GUI

Variable
ID: channelBaudRate
Value: 128k

Variable
ID: samples_per_symbol
Value: 8

Variable
ID: broadcastFreq
Value: 2.489G

Variable
ID: samp_rate
Value: 1.024M

Variable
ID: channelBW
Value: 172.8k

Variable
ID: toneLen_B
Value: 64

Variable
ID: UWLen_B
Value: 4

Variable
ID: frameNumLen_B
Value: 1

Variable
ID: gaurdLen_B
Value: 1

Variable
ID: payloadLen_B
Value: 50

Variable
ID: frameLen_B
Value: 56

Variable
ID: framesPerBurst
Value: 4

Variable
ID: nfilts
Value: 128

Variable
ID: alpha
Value: 350m

Variable
ID: rrc_taps_rx
Value: firdes.root raised ...

Variable
ID: rrc_taps_tx
Value: firdes.root raised ...

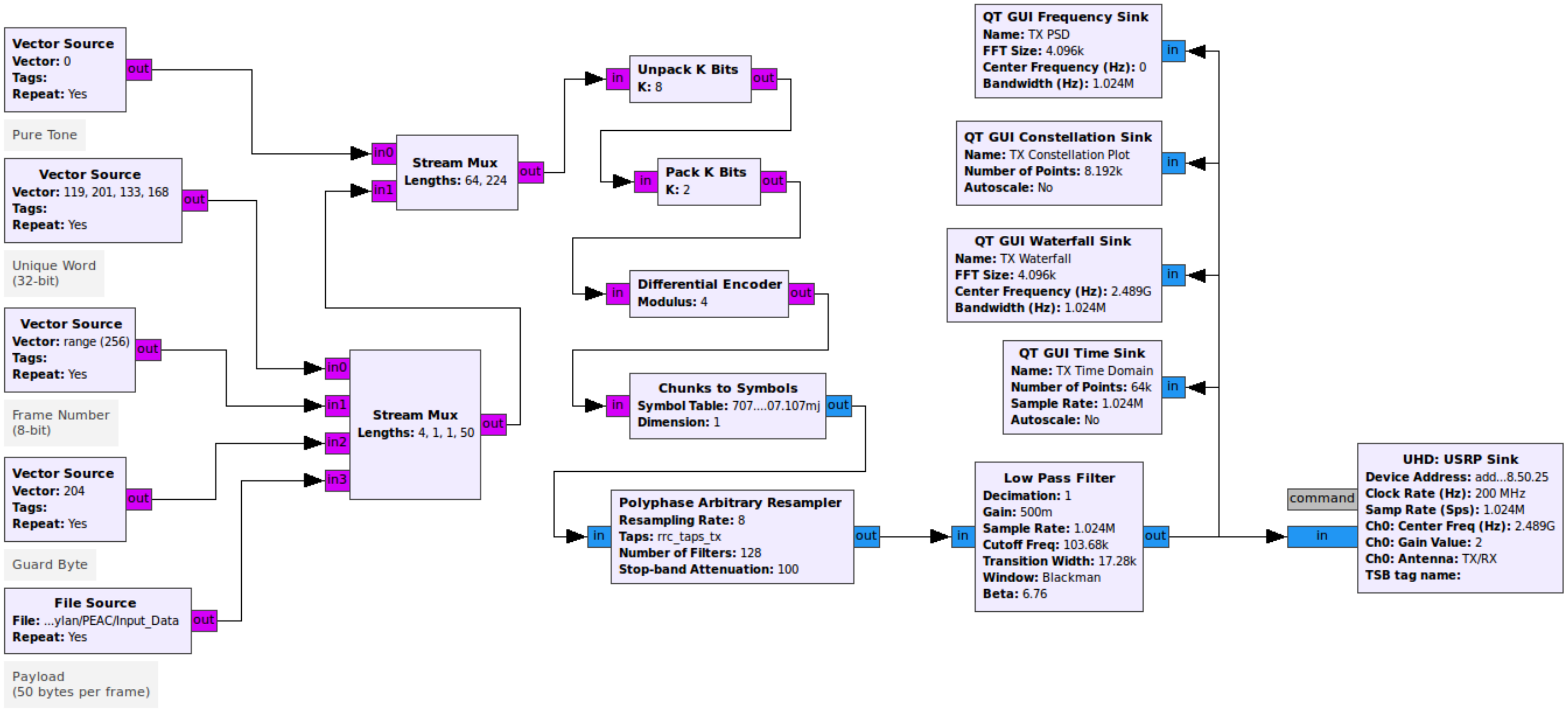
Constellation Object
ID: qpsk
Constellation Type: Variable Constellation
Symbol Map: 0, 1, 2, 3
Constellation Points: ...07mj
Rotational Symmetry: 4
Dimensionality: 1

QT GUI Tab Widget
Num Tabs: 2
Label 0: TX
Label 1: RX

QT GUI Range
ID: TX_gain
Label: TX Gain
Default Value: 2
Start: 0
Stop: 90
Step: 1

QT GUI Range
ID: RX_gain
Label: RX Gain
Default Value: 32
Start: 0
Stop: 90
Step: 1

Figure 5.2: Parameter Blocks for the QPSK OTA Flowgraph.



Vector Source
Vector: 0
Tags:
Repeat: Yes

Vector Source
Vector: 119, 201, 133, 168
Tags:
Repeat: Yes

Vector Source
Vector: range (256)
Tags:
Repeat: Yes

Vector Source
Vector: 204
Tags:
Repeat: Yes

File Source
File: ...ylan/PEAC/Input_Data
Repeat: Yes

Stream Mux
Lengths: 4, 1, 1, 50

Stream Mux
Lengths: 64, 224

Unpack K Bits
K: 8

Pack K Bits
K: 2

Differential Encoder
Modulus: 4

Chunks to Symbols
Symbol Table: 707....07.107mj
Dimension: 1

Polyphase Arbitrary Resampler
Resampling Rate: 8
Taps: rrc_taps_tx
Number of Filters: 128
Stop-band Attenuation: 100

Low Pass Filter
Decimation: 1
Gain: 500m
Sample Rate: 1.024M
Cutoff Freq: 103.68k
Transition Width: 17.28k
Window: Blackman
Beta: 6.76

QT GUI Frequency Sink
Name: TX PSD
FFT Size: 4.096k
Center Frequency (Hz): 0
Bandwidth (Hz): 1.024M

QT GUI Constellation Sink
Name: TX Constellation Plot
Number of Points: 8.192k
Autoscale: No

QT GUI Waterfall Sink
Name: TX Waterfall
FFT Size: 4.096k
Center Frequency (Hz): 2.489G
Bandwidth (Hz): 1.024M

QT GUI Time Sink
Name: TX Time Domain
Number of Points: 64k
Sample Rate: 1.024M
Autoscale: No

UHD: USRP Sink
Device Address: add...8.50.25
Clock Rate (Hz): 200 MHz
Samp Rate (Sps): 1.024M
Ch0: Center Freq (Hz): 2.489G
Ch0: Gain Value: 2
Ch0: Antenna: TX/RX
TSB tag name:

Figure 5.3: GRC Flowgraph for a Single-Carrier QPSK Transmitter.

5.2.1 Data Management

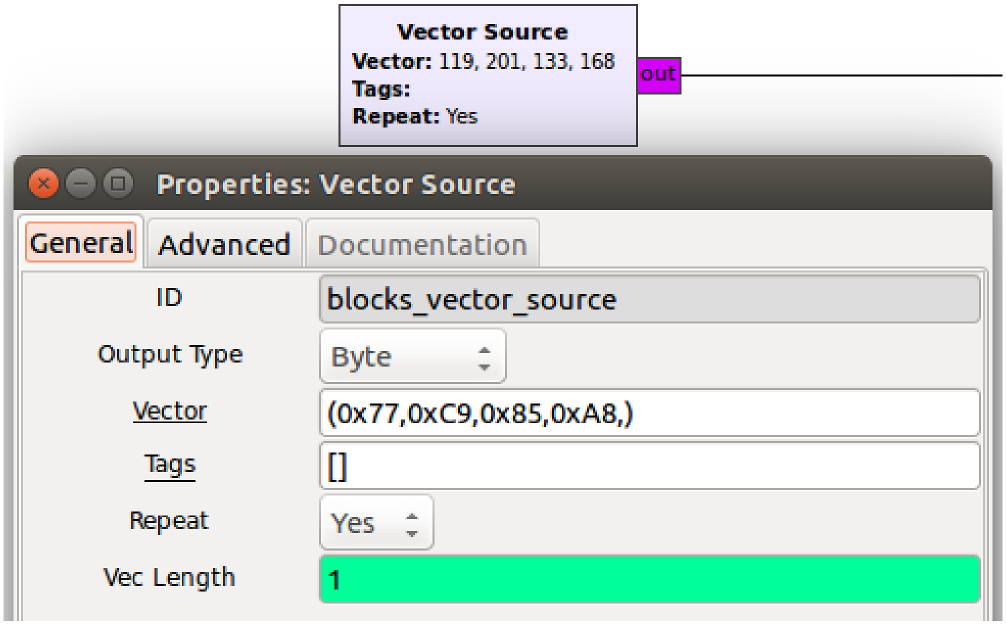
Information is often sent in a packet format to add additional information, or metadata, about the information being sent. Metadata is included in a header while the source information is sent in the payload. Even for physical layer processing, header data is useful for incorporating error correcting codes such as the Cyclic Redundancy Check (CRC) code. Additionally, headers can include a training sequence or pilot which can be used for channel estimation, carrier recovery, and phase correction.

The QPSK OTA flowgraph produces TDMA frames similar to the structure used in GSM cellular systems [37]. Starting from the left of Figure 5.3, there are five components of the packeted TDMA signal: pure tone, unique word, frame number, guard byte, and payload. The pure tone separates each packet and can be used for frequency error estimation. The unique word and frame number are incorporated in the packet header and the guard byte separates the header from the payload. The data for each of the components of the packet header is generated with a `Vector Source` block while the payload of the packet is generated with a `File Source` block. Both of these blocks are configured to repeat specified bytes indefinitely.

The components of the packets are multiplexed together using a `Stream Mux` block, then a second `Stream Mux` block is used to multiplex the packet with the tone to form a TDMA frame. The multiplexed stream of bytes are unpacked into bits and repacked into 2-bit chunks with the `Unpack K Bits` and the `Pack K Bits` blocks, respectively. The 2-bits chunks are then encoded with the `Differential Encoder` block.

Vector Source

Figure 5.4 shows the `Vector Source` block and the parameters used for the unique word in the QPSK OTA flowgraph. The output type is adjustable between the standard data types in GNU Radio (complex, float, short, int, byte). The vector parameter is specified as a tuple or array and can be either hex or decimal and is used to generate the output. Stream tags can be applied with the tags parameter. Repeat specifies whether or not the source should repeat the vector after it has reached the end of it. Vec Length specifies the length of each output stream item from this source.



Vector Source
Vector: 119, 201, 133, 168
Tags:
Repeat: Yes

Properties: Vector Source

ID: blocks_vector_source
Output Type: Byte
Vector: (0x77,0xC9,0x85,0xA8,)
Tags: []
Repeat: Yes
Vec Length: 1

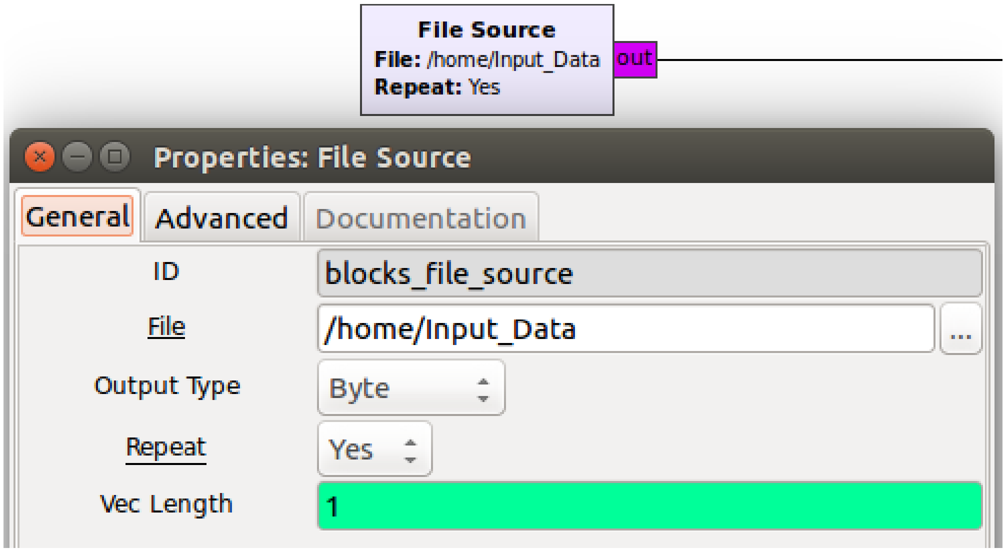
Figure 5.4: `Vector Source` Block and its relevant parameters.

File Source

Figure 5.5 shows the `File Source` block and the parameters used for the payload source in the QPSK OTA flowgraph. The file input parameter specifies a path to a file used for the output of the block. The output type, repeat, and vec length parameters are all identical to the `Vector Source` block.

Stream Mux

Figure 5.6 shows the `Stream Mux` block and the parameters used by the left-most stream mux in Figure 5.3. This block takes 4 bytes from its top input, 1 byte from the next input, 1

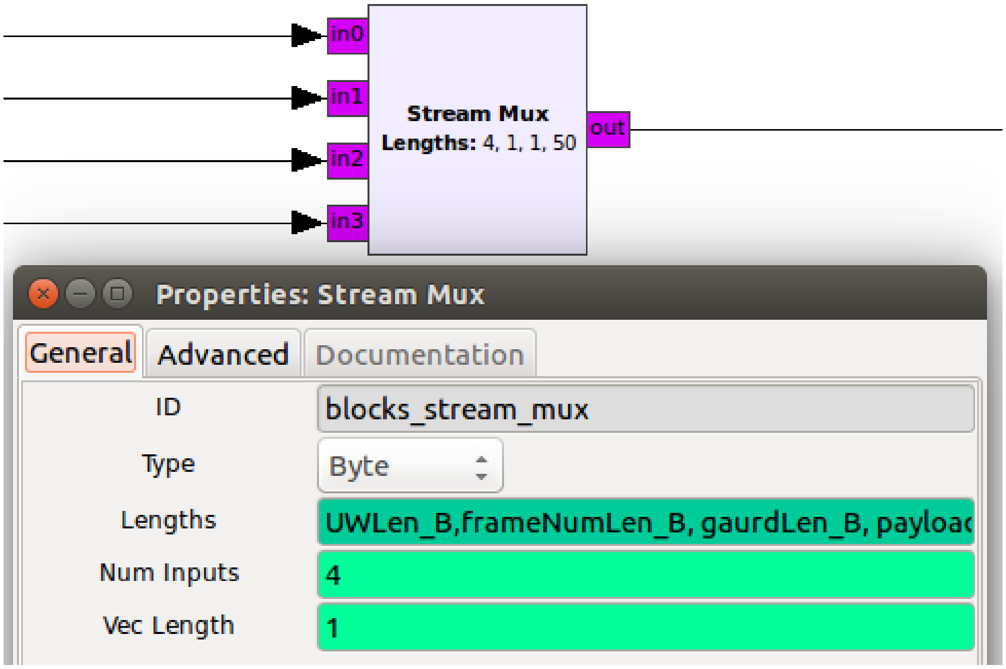


File Source
File: /home/Input_Data
Repeat: Yes

ID blocks_file_source
File /home/Input_Data
Output Type Byte
Repeat Yes
Vec Length 1

Figure 5.5: File Source Block and its relevant parameters.

byte from the following input, and 50 bytes from the bottom input. The output is 66 bytes long, and repeats as long as each of the sources feeding the inputs repeat. This behavior is based on the lengths parameter which determines the allocation of each input port in the output. Num inputs specifies the number of input ports multiplexed together by the block.

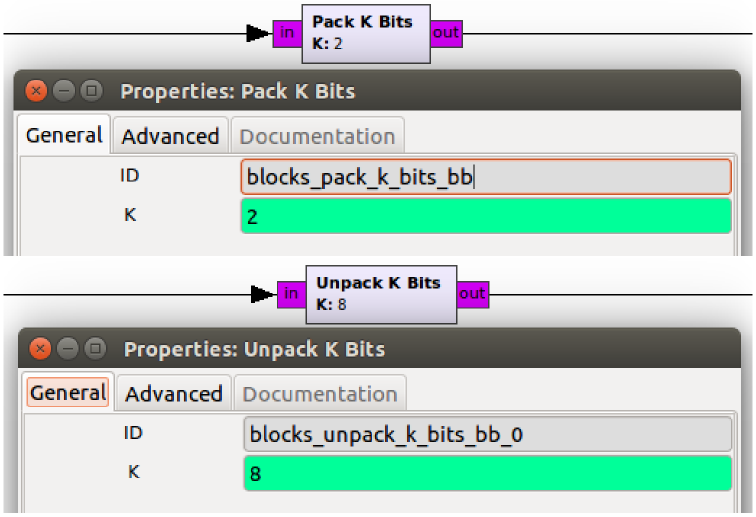


ID: blocks_stream_mux
Type: Byte
Lengths: UWLen_B,frameNumLen_B, gaurdLen_B, payloac
Num Inputs: 4
Vec Length: 1

Figure 5.6: `Stream Mux` Block and its relevant parameters.

Unpack K Bits and Pack K Bits

Figure 5.7 shows both the `Unpack K Bits` and the `Pack K Bits` blocks along with the parameters used in the transmitter of the QPSK OTA flowgraph.



ID: blocks_pack_k_bits_bb
K: 2

ID: blocks_unpack_k_bits_bb_0
K: 8

Figure 5.7: `Pack K Bits` and `Unpack K Bits` Blocks and their relevant parameters.

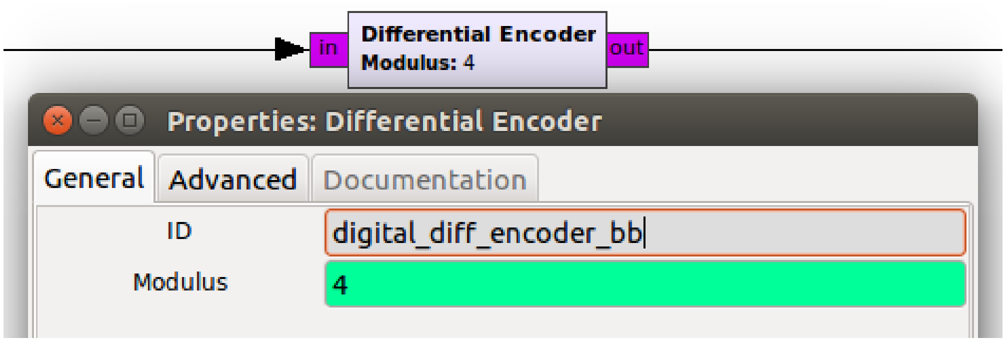
Differential Encoder

Differential coding is used to encode the source information in the change of symbol states, rather than the state itself. For example, consider the constellation diagram for QPSK in Figure 5.9. Each chunk of two bits maps to a particular constellation point, for instance, a point in the upper-right quadrant maps to 00. With differential encoding applied before this mapping, the information is now mapped to a change in the points. An example of this is that now a change from a point in the upper-right quadrant to a point in the upper-left quadrant now maps to 00. In this way, the information is relies on two constellation points instead of one.

The main benefit of Differential coding is that the source information can be received without resolving the absolute phase offset of the signal. Alternatively, a known portion of the signal, such as a training sequence, could be used to check all possible phase offsets to find the correct one.

The main drawback to differential coding is that if a single encoded symbol is received incorrectly, it will result in two incorrectly decoded symbols. If consecutive symbols rarely experience errors, this can double the BER!

Figure 5.8 shows the `Differential Encoder` block along with the parameters used in the QPSK OTA flowgraph. The modulus parameter specifies the total number of constellation points.



ID digital_diff_encoder_bb

Modulus 4

Figure 5.8: `Differential Encoder` Block and its relevant parameters.

5.2.2 Digital Modulation

Quadrature Phase-Shift Keying (QPSK) is a digital modulation technique which encodes source information into four discrete phases on a carrier signal. This is implemented by mapping chunks of two input bits to one of four complex symbols. The mapping is depicted in the constellation diagram in Figure 5.9.

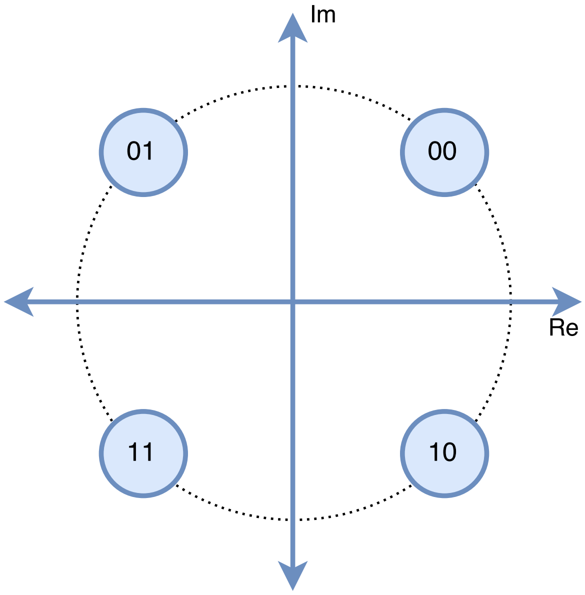
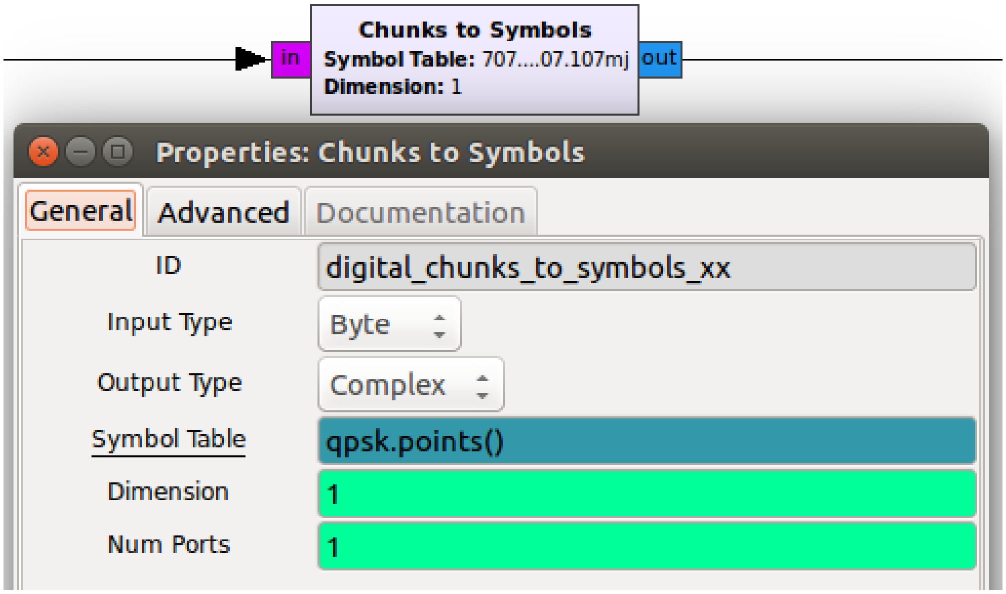


Figure 5.9: QPSK constellation plot with Gray coding.

The stream of TDMA frame bytes are unpacked into pairs of bits and mapped to QPSK symbols using the `Chunks to Symbols` block.

Chunks to Symbols

Figure 5.10 shows the `Chunks to Symbols` block and the parameters used in the QPSK OTA flowgraph. The symbol table parameter is a mapping of the input bits to the output symbols, and in this case the mapping is specified by an attribute of the `Constellation Object`.



Chunks to Symbols
Symbol Table: 707....07.107mj
Dimension: 1

ID	digital_chunks_to_symbols_xx
Input Type	Byte
Output Type	Complex
Symbol Table	qpsk.points()
Dimension	1
Num Ports	1

Figure 5.10: `Chunks to Symbols` Block and its relevant parameters.

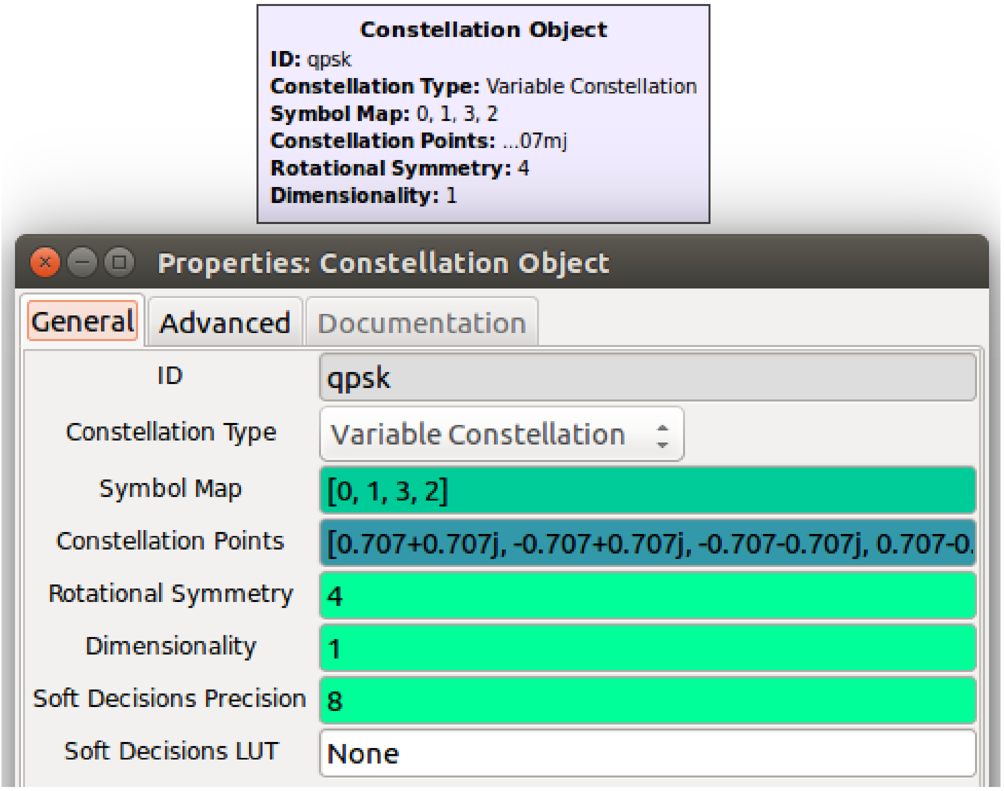
Constellation Object

Figure 5.11 shows the `Constellation Object` block and the parameters used in the QPSK OTA flowgraph. The Symbol Map and Constellation Points parameters specify the chunks of bits and complex symbols to be mapped, respectively.

5.2.3 Pulse Shaping and Resampling

The QPSK symbols are then pulse shaped and up-sampled in the `Polyphase Arbitrary Resampler` block. Additionally, the low-pass filter used in the `Polyphase Arbitrary Resampler` block is used to matched filter the baseband data with a Root Raised Cosine (RRC) filter. The RRC drop-off is determined by the alpha parameter.

Next, the upsampled QPSK samples are low-pass filtered to eliminate the high frequency distortion resulting from the upsampling and matched filtering.



Constellation Object
ID: qpsk
Constellation Type: Variable Constellation
Symbol Map: 0, 1, 3, 2
Constellation Points: ...07mj
Rotational Symmetry: 4
Dimensionality: 1

ID	qpsk
Constellation Type	Variable Constellation
Symbol Map	[0, 1, 3, 2]
Constellation Points	[0.707+0.707j, -0.707+0.707j, -0.707-0.707j, 0.707-0.
Rotational Symmetry	4
Dimensionality	1
Soft Decisions Precision	8
Soft Decisions LUT	None

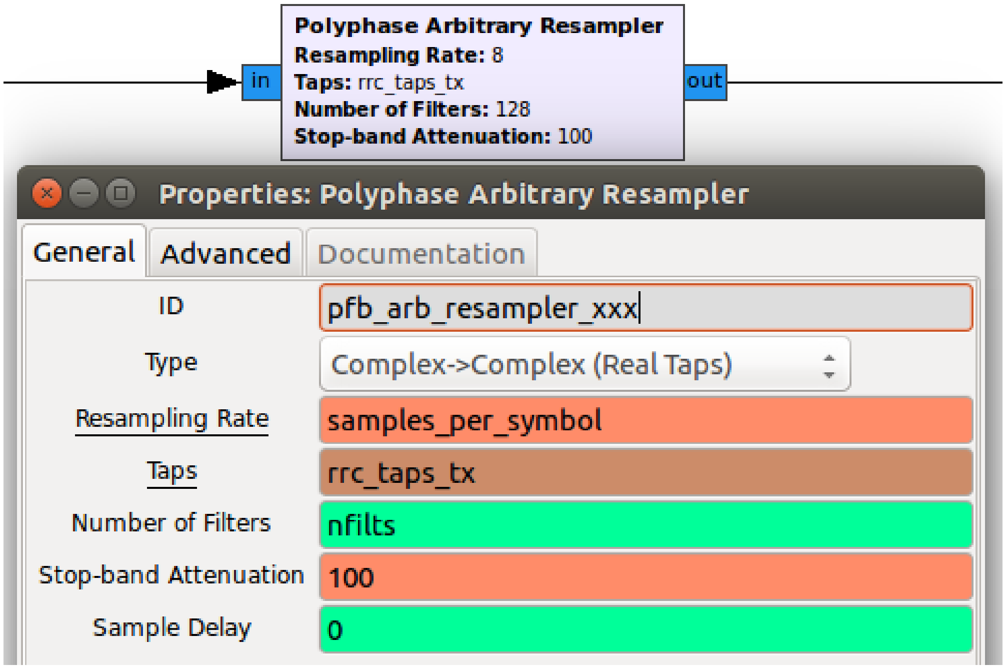
Figure 5.11: `Constellation Object` Block and its relevant parameters.

Polyphase Arbitrary Resampler

Figure 5.12 shows the `Polyphase Arbitrary Resampler` block and the parameters used in the QPSK OTA flowgraph. The algorithm behind this block is derived from Section 7.5 of [38].

Low-Pass Filter

Figure 5.13 shows the `Low Pass Filter` block and the parameters used in the QPSK OTA flowgraph. The gain, cutoff freq, and transition width parameters of the filter can be directly specified. Additionally, different windowing functions can be applied to the filter such as Blackman, Rectangular, and Hamming. The filter can be used to simultaneously decimate samples by a factor specified in the decimation parameter.



Polyphase Arbitrary Resampler
Resampling Rate: 8
Taps: rrc_taps_tx
Number of Filters: 128
Stop-band Attenuation: 100

Properties: Polyphase Arbitrary Resampler

ID	pfb_arb_resampler_xxx
Type	Complex->Complex (Real Taps)
Resampling Rate	samples_per_symbol
Taps	rrc_taps_tx
Number of Filters	nfilts
Stop-band Attenuation	100
Sample Delay	0

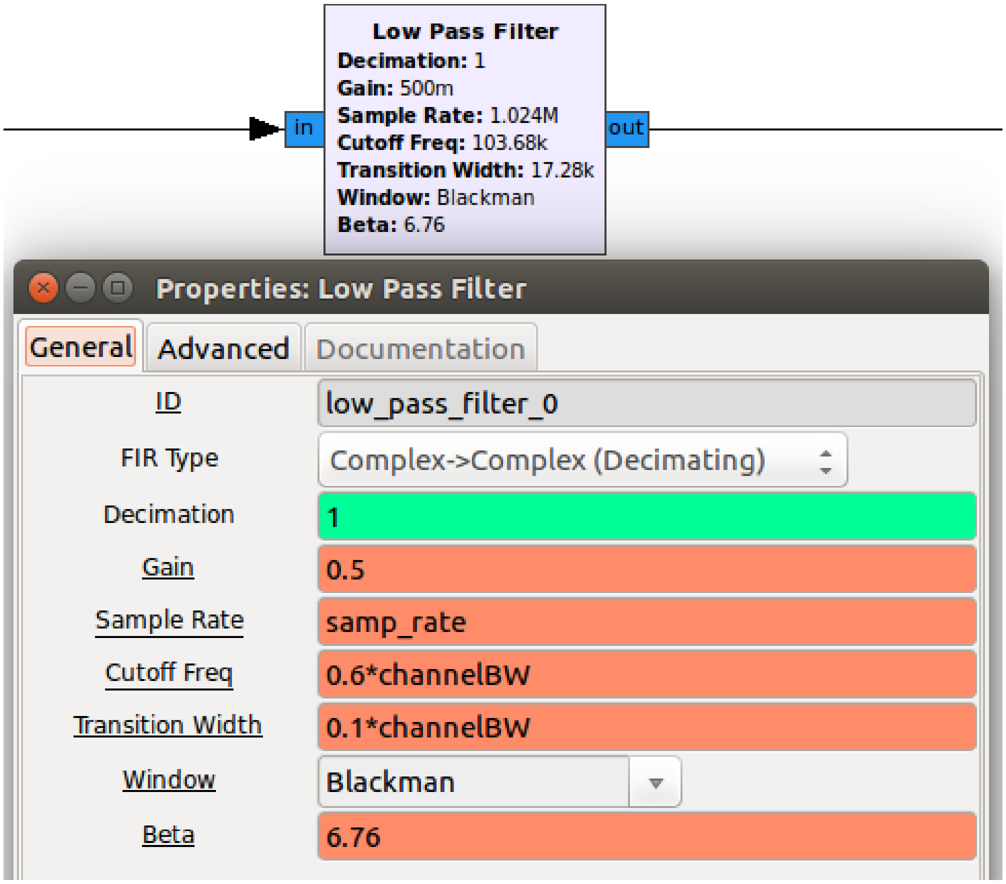
Figure 5.12: `Polyphase Arbitrary Resampler` block and its relevant parameters.

5.2.4 RF Front-End

Finally, the matched-filtered and upsampled baseband symbols are transmitted to the `UHD: USRP Sink` block. The Universal Hardware Driver (UHD) is device driver provided by Ettus Research which interfaces with all USRPs. The complex samples received by the `UHD: USRP Sink` block are upconverted and transmitted by the USRP.

UHD: USRP Sink

Figure 5.14 displays the necessary parameters to specify in order to send data generated in GNU Radio to a USRP using the UHD. On the left are general parameters which deal with connection and configuration of the device. The device address of the USRP is either a serial number if connecting over USB or an IP address for a networked connection and can be set to multiple addresses. The number of motherboards and channels associated with this source are also specified. The synchronization and clock sources are specified in



Low Pass Filter
Decimation: 1
Gain: 500m
Sample Rate: 1.024M
Cutoff Freq: 103.68k
Transition Width: 17.28k
Window: Blackman
Beta: 6.76

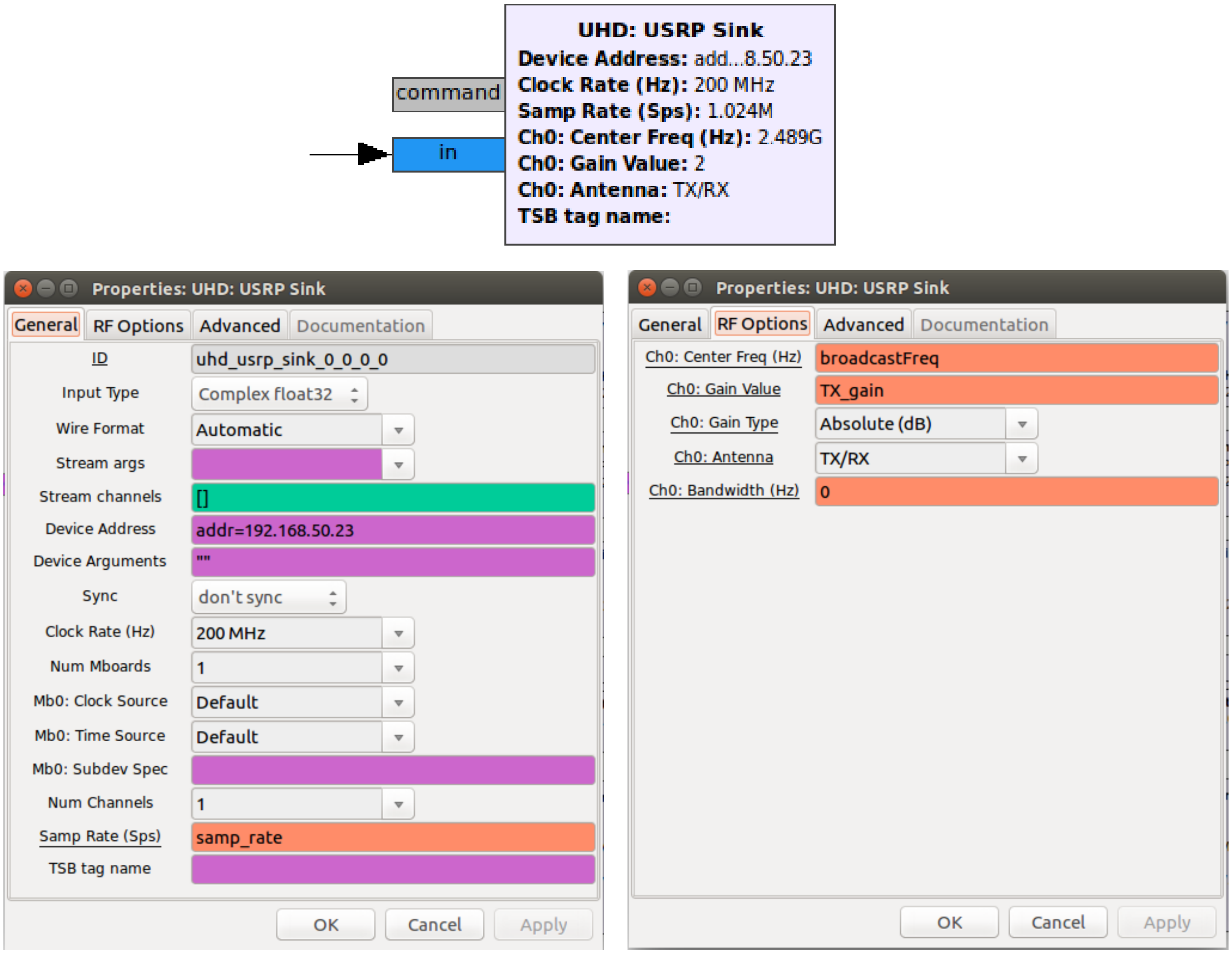
Properties: Low Pass Filter

ID	low_pass_filter_0
FIR Type	Complex->Complex (Decimating)
Decimation	1
Gain	0.5
Sample Rate	samp_rate
Cutoff Freq	0.6*channelBW
Transition Width	0.1*channelBW
Window	Blackman
Beta	6.76

Figure 5.13: `Polyphase Arbitrary Resampler` block and its relevant parameters.

the case where an external clock source or synchronizer is being used.

On the right of Figure 5.14 are the RF parameters. The center frequency of the RF chain, gain applied in the RF circuit, and antenna port are all specified in this tab. The channel bandwidth is set to 0 to use the default filter settings.



UHD: USRP Sink
Device Address: add...8.50.23
Clock Rate (Hz): 200 MHz
Samp Rate (Sps): 1.024M
Ch0: Center Freq (Hz): 2.489G
Ch0: Gain Value: 2
Ch0: Antenna: TX/RX
TSB tag name:

Properties: UHD: USRP Sink

ID	uhd_usrp_sink_0_0_0_0
Input Type	Complex float32
Wire Format	Automatic
Stream args	
Stream channels	[]
Device Address	addr=192.168.50.23
Device Arguments	""
Sync	don't sync
Clock Rate (Hz)	200 MHz
Num Mboards	1
Mb0: Clock Source	Default
Mb0: Time Source	Default
Mb0: Subdev Spec	
Num Channels	1
Samp Rate (Sps)	samp_rate
TSB tag name	

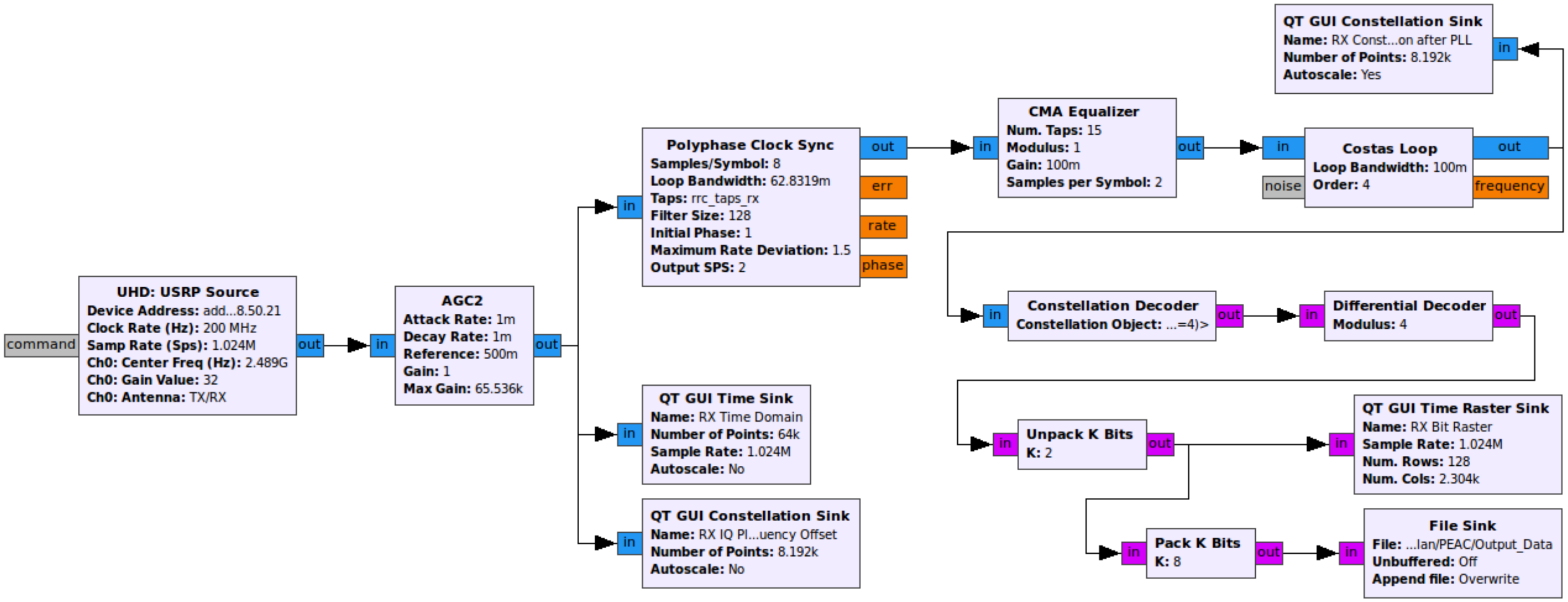
Properties: UHD: USRP Sink

Ch0: Center Freq (Hz)	broadcastFreq
Ch0: Gain Value	TX_gain
Ch0: Gain Type	Absolute (dB)
Ch0: Antenna	TX/RX
Ch0: Bandwidth (Hz)	0

Figure 5.14: `UHD: USRP Sink` Block and its relevant parameters.

5.3 QPSK Receiver

The GRC flowgraph in Figure 5.15 implements the corresponding receiver for the single-carrier OTA QPSK system.



UHD: USRP Source
Device Address: add...8.50.21
Clock Rate (Hz): 200 MHz
Samp Rate (Sps): 1.024M
Ch0: Center Freq (Hz): 2.489G
Ch0: Gain Value: 32
Ch0: Antenna: TX/RX

AGC2
Attack Rate: 1m
Decay Rate: 1m
Reference: 500m
Gain: 1
Max Gain: 65.536k

Polyphase Clock Sync
Samples/Symbol: 8
Loop Bandwidth: 62.8319m
Taps: rrc_taps_rx
Filter Size: 128
Initial Phase: 1
Maximum Rate Deviation: 1.5
Output SPS: 2

QT GUI Time Sink
Name: RX Time Domain
Number of Points: 64k
Sample Rate: 1.024M
Autoscale: No

QT GUI Constellation Sink
Name: RX IQ Pl...uency Offset
Number of Points: 8.192k
Autoscale: No

CMA Equalizer
Num. Taps: 15
Modulus: 1
Gain: 100m
Samples per Symbol: 2

Costas Loop
Loop Bandwidth: 100m
Order: 4

QT GUI Constellation Sink
Name: RX Const...on after PLL
Number of Points: 8.192k
Autoscale: Yes

Constellation Decoder
Constellation Object: ...=4)>

Differential Decoder
Modulus: 4

Unpack K Bits
K: 2

QT GUI Time Raster Sink
Name: RX Bit Raster
Sample Rate: 1.024M
Num. Rows: 128
Num. Cols: 2.304k

Pack K Bits
K: 8

File Sink
File: ...lan/PEAC/Output_Data
Unbuffered: Off
Append file: Overwrite

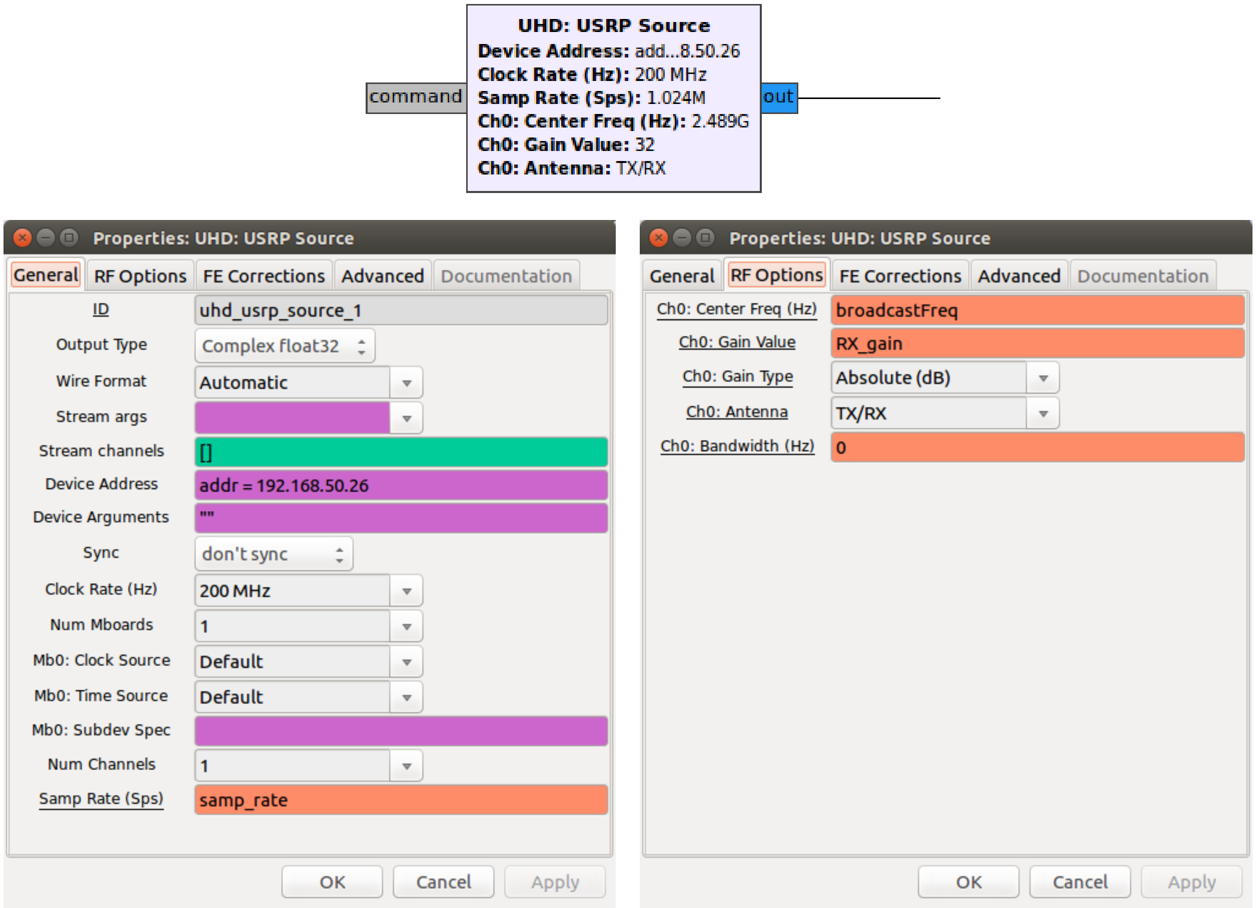
Figure 5.15: GRC Flowgraph for a Single-Carrier QPSK Receiver.

5.3.1 RF Front-End

Starting from the left, the signals received by the USRP are processed and communicated to GNU Radio through the `UHD: USRP Source` block.

UHD: USRP Source

Figure 5.16 shows the `UHD: USRP Source` block and the parameters used in the QPSK OTA flowgraph. The parameters for the `UHD: USRP Source` block are identical to the `UHD: USRP Sink` block used in the transmitter, except that the device address now references the receiving USRP and a different gain is applied.



UHD: USRP Source
Device Address: add...8.50.26
Clock Rate (Hz): 200 MHz
Samp Rate (Sps): 1.024M
Ch0: Center Freq (Hz): 2.489G
Ch0: Gain Value: 32
Ch0: Antenna: TX/RX

Properties: UHD: USRP Source

ID	uhd_usrp_source_1
Output Type	Complex float32
Wire Format	Automatic
Stream args	
Stream channels	[]
Device Address	addr = 192.168.50.26
Device Arguments	""
Sync	don't sync
Clock Rate (Hz)	200 MHz
Num Mboards	1
Mb0: Clock Source	Default
Mb0: Time Source	Default
Mb0: Subdev Spec	
Num Channels	1
Samp Rate (Sps)	samp_rate

Properties: UHD: USRP Source

Ch0: Center Freq (Hz)	broadcastFreq
Ch0: Gain Value	RX_gain
Ch0: Gain Type	Absolute (dB)
Ch0: Antenna	TX/RX
Ch0: Bandwidth (Hz)	0

Figure 5.16: `UHD: USRP Source` Block and its relevant parameters.

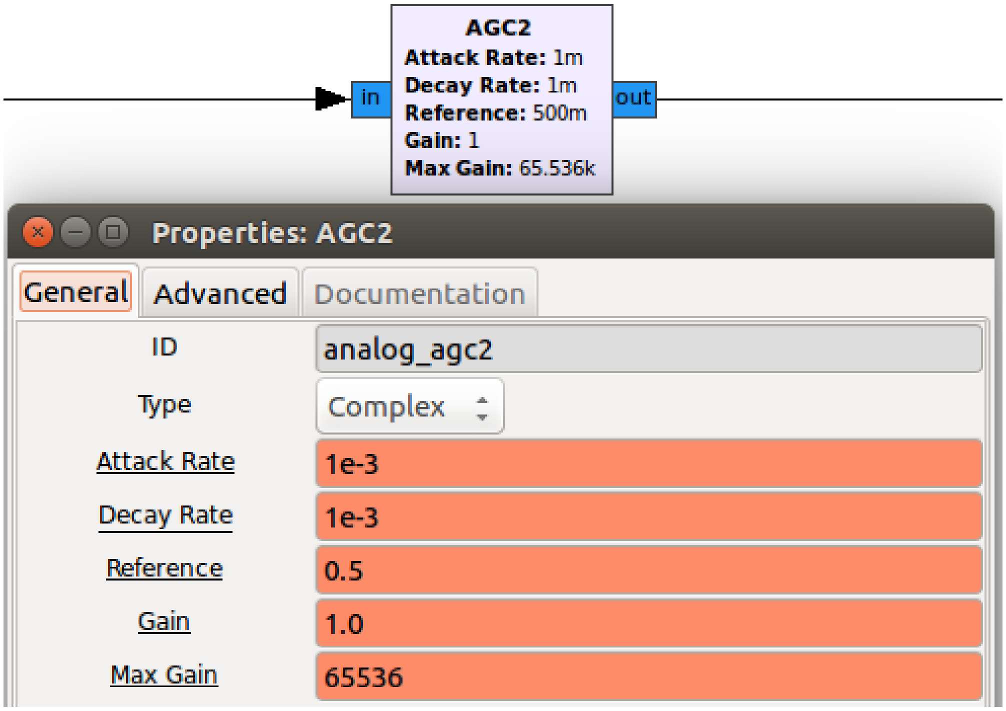
5.3.2 Automatic Gain Control

The received samples are passed through the `AGC2` block which implements an Automatic Gain Control (AGC) closed feedback loop.

AGC2

Figure 5.17 shows the AGC2 block and the parameters used in the QPSK OTA flowgraph. The AGC loop attempts to keep the output a constant amplitude when the input amplitude fluctuates. The Reference parameter specifies the target amplitude for the output while the attack and decay Rates determine how quickly the loop adjusts the outputs. Specifically, the attack rate determines the adjustment step size for decreasing the gain and the decay rate determines the adjustment step size for increasing the gain. The max gain parameter specifies the maximum gain that can be provided by the AGC.

For an AGC with a high attack rate and low Decay rate, the output will quickly decrease when a large signal appears and will slowly increase when that signal goes away.



AGC2
Attack Rate: 1m
Decay Rate: 1m
Reference: 500m
Gain: 1
Max Gain: 65.536k

ID	analog_agc2
Type	Complex
Attack Rate	1e-3
Decay Rate	1e-3
Reference	0.5
Gain	1.0
Max Gain	65536

Figure 5.17: AGC2 Block with Parameters Specified for OTA QPSK flowgraph.

Control Loops in GNU Radio

[7] describes the way that control loops are implemented in GNU Radio. Control loops are used in many elements of receive processing including clock recovery, blind equalizer, and phase recovery (Phase Lock Loop). These control loops can be modeled as shown in Figure 5.18.

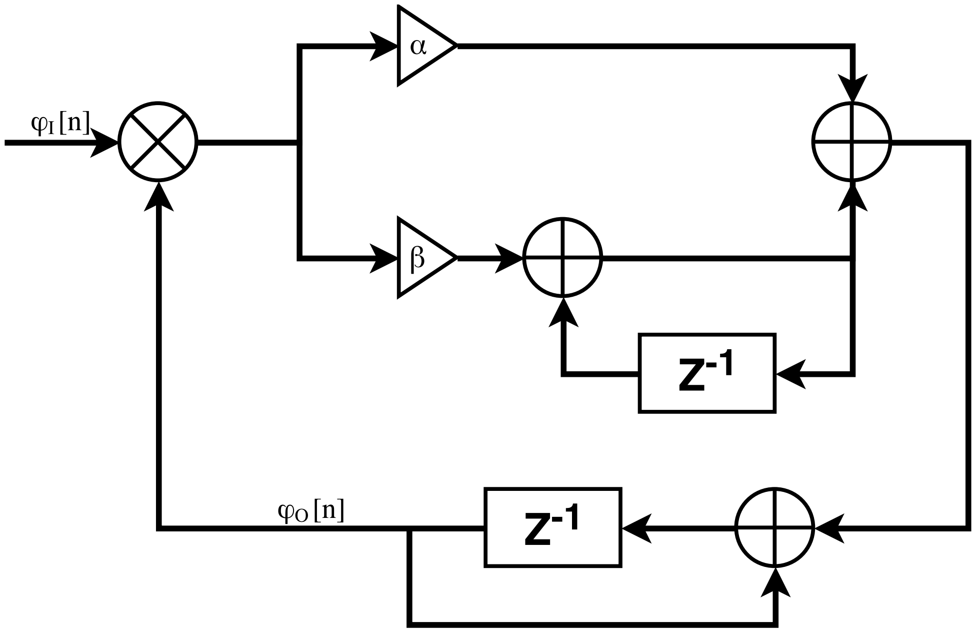


Figure 5.18: Control Loop with input ϕ_I and output ϕ_O [7].

The path gains, α and β are used to adjust the step size of the loop's convergence in frequency and phase, respectively. This relationship can be expressed as

$$f = f + \alpha(error)$$

$$\phi = \phi + f + \beta(error)$$

To get a more intuitive understanding of the loop gains, they can be related to the damping ratio of the loop ζ and the loop bandwidth θ_n

$$\alpha = \frac{4\zeta\theta_n}{1 + 2\zeta\theta_n + \theta_n^2}$$

$$\beta = \frac{4\theta_n^2}{1 + 2\zeta\theta_n + \theta_n^2}$$

A derivation of this relationship can be found in [7].

A real control system (one that experiences finite loss) can exist in one of three states:

underdamped, overdamped, and critically damped. In the underdamped case, the losses on the system are small enough that the system will overshoot it's original state and loose energy slowly through each oscillation until it returns to that state. If the losses in the system are large, the system will return to its original state without ever overshooting it's original state - this is the overdamped case. Finally, a critically damped system lies between the underdamped and overdamped cases and will fail to overshoot or complete an oscillation before returning to its original state.

The damping ratio is defined as the ratio of the damping of the system to the critical damping case. In most applications, a critically damped system is desirable because it converges to the original state quicker than the other cases of damping. Therefore, the loop blocks in GNU Radio have the damping ratio of all loop blocks internally set to $\zeta = \frac{\sqrt{2}}{2}$ so that the only loop gain parameter that needs to be adjusted is the loop bandwidth.

[7] recommends setting the loop bandwidth between $\frac{2\pi}{200}$ and $\frac{2\pi}{100}$. The relationship between the loop gains and the loop bandwidth for a critically damped control loop is shown in Figure 5.19.

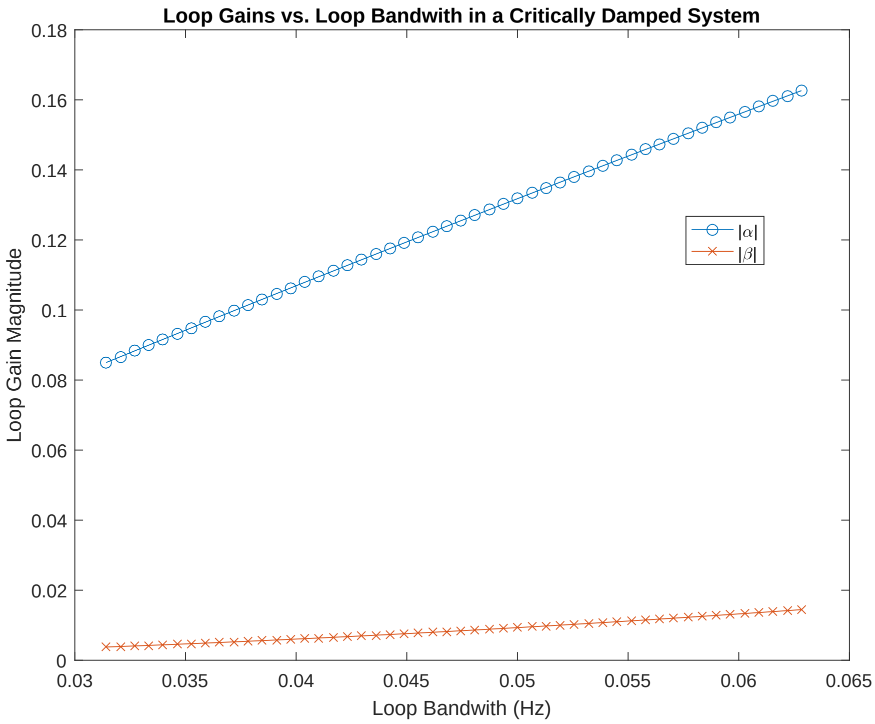


Figure 5.19: Control loop gains vs. loop bandwidth in a critically damped system.

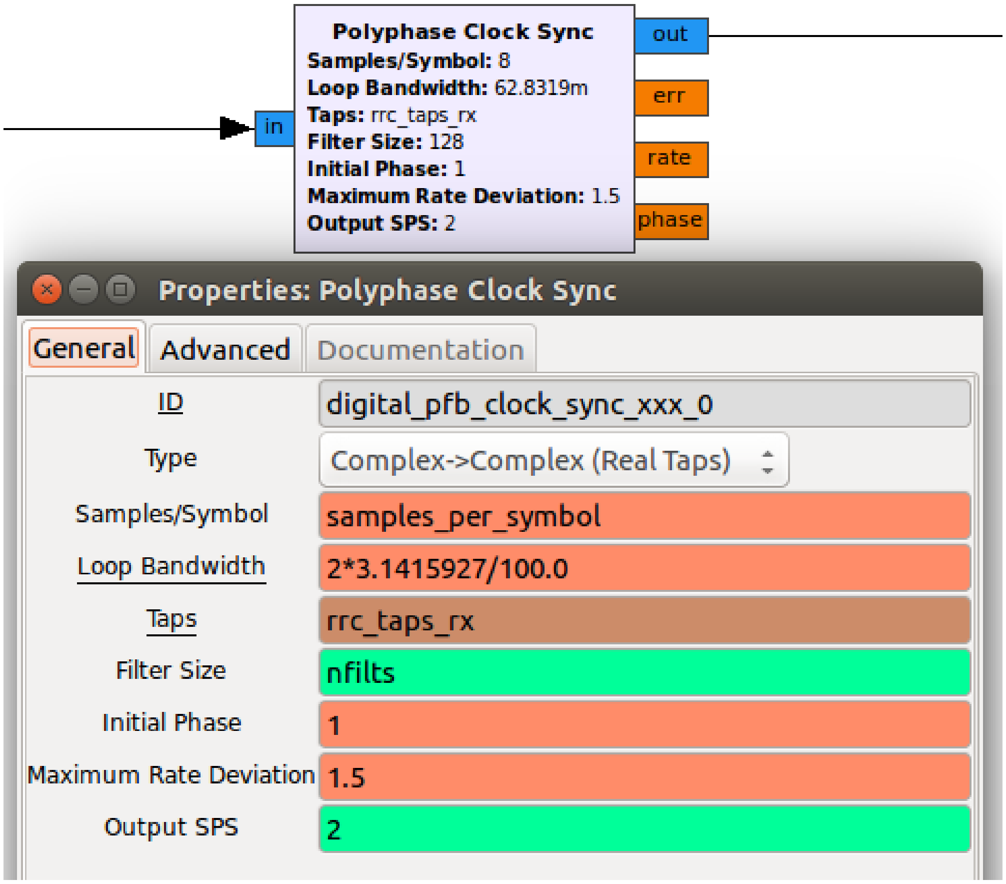
Figure 5.19 demonstrates that as the loop bandwidth is increased, the loop gains will increase. This will increase the step size for the loop which results in a quicker, but less accurate convergence. This value is often adjusted empirically to find the optimal value, or just left at a value in the recommended range.

5.3.3 Timing Recovery and Matched Filtering

The next step in the receiver chain is to eliminate any timing offset between the clock at receiver and transmitter. Next, the receiver applies a RRC matched filter to eliminate Inter Symbol Interference (ISI).

Polyphase Clock Sync

Figure 5.21 shows the `Polyphase Clock Sync` block and the parameters used in the QPSK OTA flowgraph. This block uses the polyphase filterbank clock recovery algorithm described in section 13.3 of [38]. This block first recovers timing, applies a matched filter, and resamples to a new number of samples per symbol specified by the output SPS parameter.



Polyphase Clock Sync
Samples/Symbol: 8
Loop Bandwidth: 62.8319m
Taps: rrc_taps_rx
Filter Size: 128
Initial Phase: 1
Maximum Rate Deviation: 1.5
Output SPS: 2

ID: digital_pfb_clock_sync_xxx_0
Type: Complex->Complex (Real Taps)
Samples/Symbol: samples_per_symbol
Loop Bandwidth: 2*3.1415927/100.0
Taps: rrc_taps_rx
Filter Size: nfilts
Initial Phase: 1
Maximum Rate Deviation: 1.5
Output SPS: 2

Figure 5.20: `Polyphase Clock Sync` Block with Parameters Specified for OTA QPSK flowgraph.

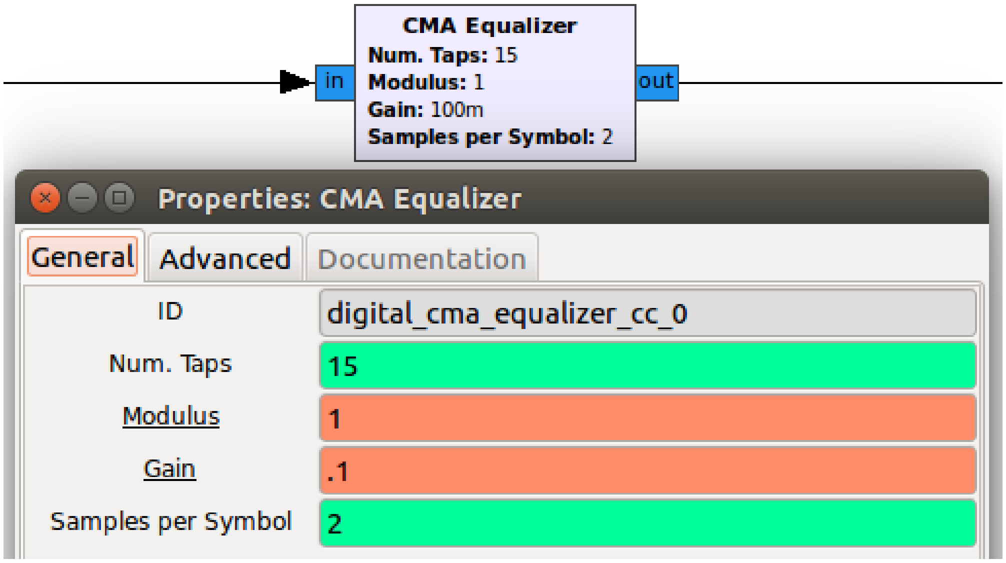
5.3.4 Equalization

Once the sample timing has been recovered, an equalization algorithm is applied to undo any fading effects introduced by the channel. Digital equalizers come in many different shapes and sizes. Blind equalizers know nothing about the constellation or the channel, but must make other assumptions about the signal such as a constant modulus. Adaptive equalizers minimize the error between the received signal and an expected value generated with statistical knowledge of the transmit signal. Adaptive equalizers don't need to know anything about the channel, but will require an initial guess in order to begin converging to an estimate of the transmit signal. If the received signal is too degraded when it reaches an adaptive equalizer, it may not converge to the correct value. For this reason, it is common to improve the received signal with frequency recovery loops and blind equalizers to coarsely acquire the signal before sending it to an adaptive equalizer for fine tuning.

Of course, if an accurate estimate of the channel is available, a data-aided equalization approach can be applied directly to the received signal with no assumptions about the transmitted signal.

CMA Equalizer

The equalization applied in the QPSK OTA flowgraph is the Constant Modulus Algorithm (CMA). CMA is a blind equalization technique which requires the transmitted modulation points to have equal magnitude, also known as a constant modulus. The modulus parameter specifies the assumed modulus of the transmit signal, which will be the magnitude level that the equalizer tries to fix the symbols to. The modulus is set to 1 so that the output magnitude is normalized.



CMA Equalizer
Num. Taps: 15
Modulus: 1
Gain: 100m
Samples per Symbol: 2

Properties: CMA Equalizer

ID	digital_cma_equalizer_cc_0
Num. Taps	15
Modulus	1
Gain	.1
Samples per Symbol	2

Figure 5.21: `CMA Equalizer` Block with Parameters Specified for OTA QPSK flowgraph.

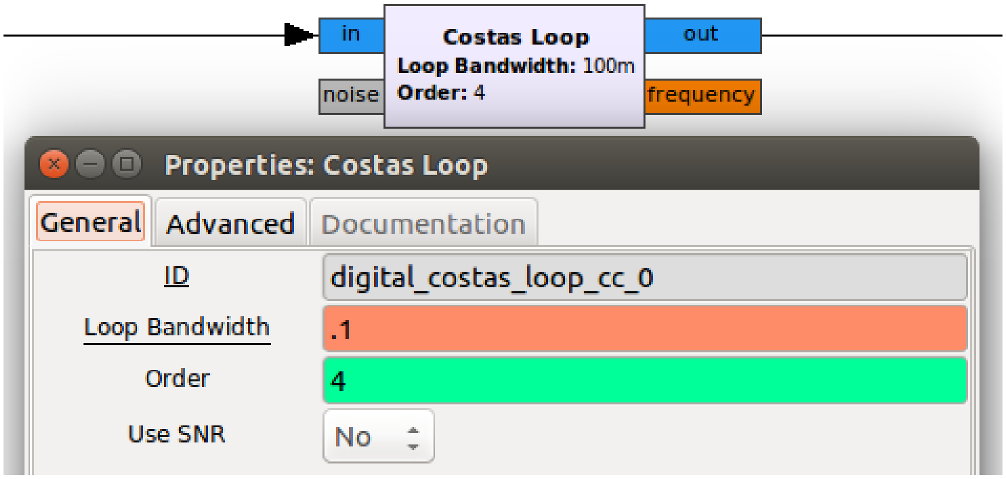
5.3.5 Carrier Frequency Recovery

When two isolated radios communicate OTA, they attempt to tune their Local Oscillators (LOs) to the same frequency. Unfortunately, there will usually be an offset in LO frequencies, even if both radios are "tuned" to the same frequency. When down-converting the signal at the receiver, this error results in a residual carrier persisting in the baseband signal. To adjust this out, the receiver can determine an estimate of the transmitter's true LO frequency and undo the carrier offset accordingly. If statistical information about the transmit signal is available, such as its constellation, then a control loop can be used to minimize the error without an estimate of the carrier.

Costas Loop

A Costas Loop is another control loop, catagorized as a Phase Lock Loop (PLL), which can be used with PSK signals to minimize the phase error with knowledge of the transmitted constellation. This loop works by measuring the symbol value at baud centers and comparing it to the closest ideal constellation point. The phase difference between the measured point and the ideal point is considered the error and is used to adjust the frequency of the baseband signal.

Figure 5.22 shows the `Costas Loop` block and the parameters used in the QPSK OTA flowgraph. The order parameter can be set to 2, 4, and 8 to work with BPSK, QPSK, and 8-PSK, respectively.



Costas Loop
Loop Bandwidth: 100m
Order: 4

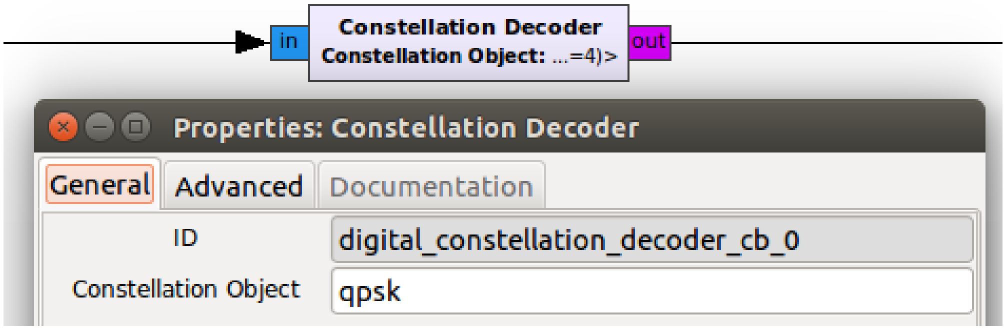
Properties: Costas Loop

ID	digital_costas_loop_cc_0
Loop Bandwidth	.1
Order	4
Use SNR	No

Figure 5.22: Costas Loop Block with Parameters Specified for OTA QPSK flowgraph.

5.3.6 Digital Demodulation

After timing recovery, equalization, and carrier frequency offset correction, the symbols are ready to be demodulated. The Constellation Decoder block shown in Figure 5.23 implements the mapping of symbols to chunks of bits. The symbol mapping is specified in the same `Constellation Object` block that the `Chunks to Symbols` block used to modulate the source information.


in
Constellation Decoder
Constellation Object: ...=4)>
out
Properties: Constellation Decoder
General
Advanced
Documentation

ID: digital_constellation_decoder_cb_0
Constellation Object: qpsk

Figure 5.23: `Constellation Decoder` Block with Parameters Specified for OTA QPSK flowgraph.

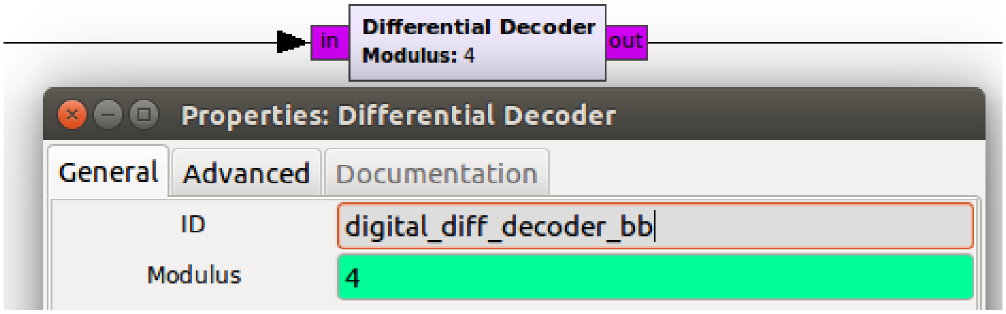
5.3.7 Data Management

After demodulation, the 2-bit chunks are run through the `Differential Decoder` block to undo the Differential coding applied at the transmitter.

The decoded chunks are unpacked into bits and repacked into bytes using the `Unpack K Bits` and the `Pack K Bits` blocks, respectively. The bits are feed into a `Time Raster` Plot and the bytes are feed into a `File Sink` to save the received data for post-processing.

Differential Decoder

Figure 5.24 shows the `Differential Decoder` block and the parameters used in the QPSK OTA flowgraph.

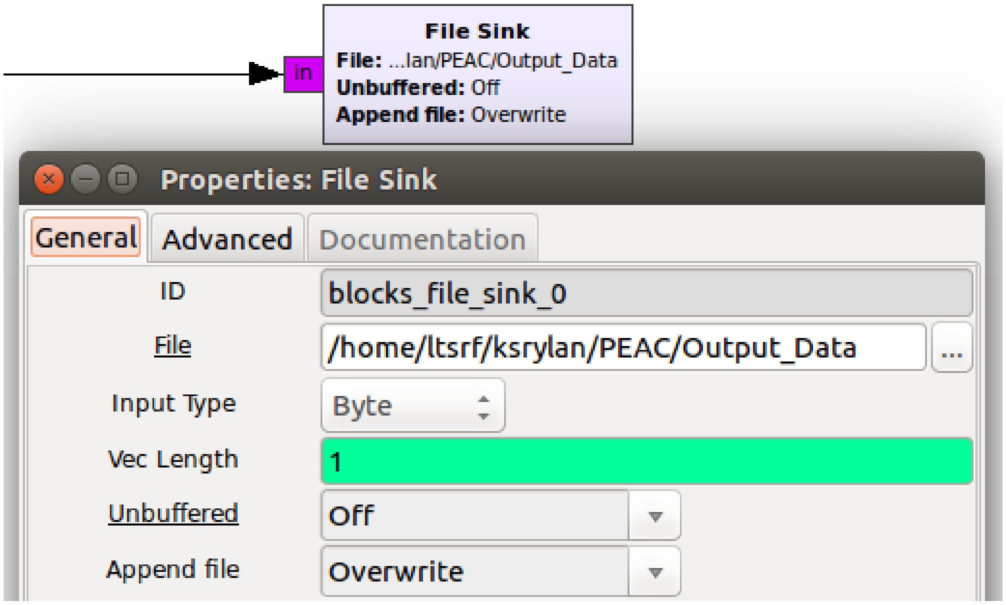


ID: digital_diff_decoder_bb
Modulus: 4

Figure 5.24: `Differential Decoder` Block with Parameters Specified for OTA QPSK flowgraph.

File Sink

Figure 5.25 shows the `File Sink` block and the parameters used in the QPSK OTA flowgraph.



File Sink
File: ...lan/PEAC/Output_Data
Unbuffered: Off
Append file: Overwrite

ID: blocks_file_sink_0
File: /home/ltsrf/ksrylan/PEAC/Output_Data
Input Type: Byte
Vec Length: 1
Unbuffered: Off
Append file: Overwrite

Figure 5.25: `File Sink` Block with Parameters Specified for OTA QPSK flowgraph.

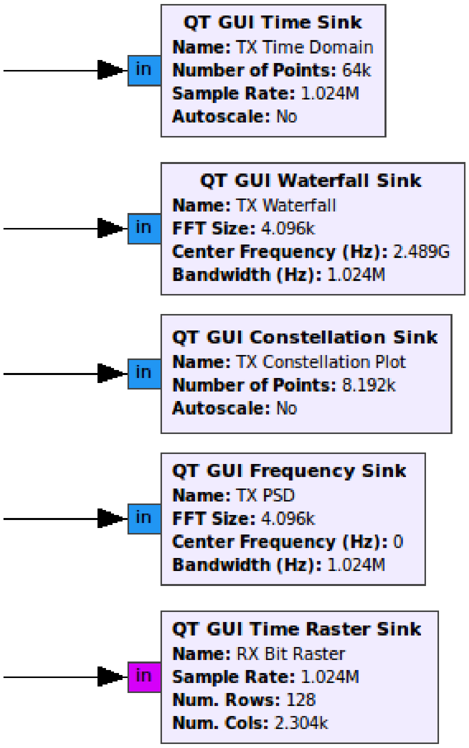
5.3.8 GUI and Output Comparison

The GUI supported by GNU Radio allows several types of plots that can be used to observe different points of the processing in real time. Additionally, parameters can be set to be adjustable from the GUI so that variables such as transmitter gain can be adjusted while operating the SDR.

GUI Blocks

Five GUI blocks used in the design of the OTA systems in this thesis are the `Time Sink`, `Waterfall Sink`, `Constellation Sink`, `Frequency Sink`, and `Time Raster`

Sink shown in Figure 5.26.



QT GUI Time Sink
Name: TX Time Domain
Number of Points: 64k
Sample Rate: 1.024M
Autoscale: No

QT GUI Waterfall Sink
Name: TX Waterfall
FFT Size: 4.096k
Center Frequency (Hz): 2.489G
Bandwidth (Hz): 1.024M

QT GUI Constellation Sink
Name: TX Constellation Plot
Number of Points: 8.192k
Autoscale: No

QT GUI Frequency Sink
Name: TX PSD
FFT Size: 4.096k
Center Frequency (Hz): 0
Bandwidth (Hz): 1.024M

QT GUI Time Raster Sink
Name: RX Bit Raster
Sample Rate: 1.024M
Num. Rows: 128
Num. Cols: 2.304k

Figure 5.26: GUI blocks used in GNU Radio.

5.4 Testbed Design

This testbed was designed to be flexibly used not only for the single-carrier applications described in this thesis, but also for experimentation with wideband waveforms such as OFDM. Such experiments require a large effective system bandwidth, reflected in the maximum sampling rate acheivable by the radio. The system bandwidth is limited by three main factors: Analog Bandwidth, FPGA Processing Bandwidth, and Host Bandwidth.

Analog Bandwidth is defined as the 3 dB bandwidth between the RF port and the IF interface in an RF channel. This bandwidth is limited by the IF filter on the USRP daughterboards [9]. This testbed uses UBX-160 daugtherboard which have an analog bandwidth of 160 MHz.

FPGA Processing Bandwidth is the maximum sampling rate that can be provided by the ADCs and DACs USRP motherboard. For a USRP X310, the ADC can sample at 200 million samples per second (MSps) and the DAC can sample at 800 MSps for 16-bit IQ samples [9]. The FPGA processing bandwidth is defined as the limiting factor, so the overall rate for the X310 is considered to be set by the ADC at 200 MSps. To compare this to the analog bandwidth set by the daughterboard, consider the effective analog bandwidth of the ADC to be 80% of the Nyquist bandwidth corresponding to the rate [39]

$$BW_{base} = B_N * .80 = (f_s/2) * .80 = (200/2) * .80 = 80\text{MHz}.$$

The Analog Bandwidth of the RF circuit is calculated for a passband signal,

$$BW_{pass} = BW_{base} * 2 = 160\text{MHz}.$$

Not surprisingly, the top performing daughterboard was designed to maximize the FPGA Bandwidth available at the motherboard. Therefore, the total bandwidth of the device is 160 MHz which corresponds to a data rate of 200 MSps.

The Host Bandwidth is the maximum data rate between the FPGA in the USRP and the host machine. Ideally, the network should be designed such that the limiting factor for sampling rate is the device connected to it. Table 5.4 shows various interfaces and their corresponding sample rates for 16-bit IQ samples [9].

The only two options that can match the device's bandwidth are 10 Gigabit Ethernet (GigE) and a 4-lane PCI Express connection. To expand the network to multiple devices and hosts, 10 GigE was chosen for the interface. A diagram of the network with throughput speeds is shown in Figure 5.27.

Interface	**Sample Rate (MSps for 16-bit IQ)**	**Half/Full Duplex**
USB 2.0	8	Half Duplex
USB 3.0	61.44	Half Duplex
Gigabit Ethernet	25	Full Duplex
10 Gigabit Ethernet	>200	Full Duplex
PCI-Express (4-lane)	>200	Full Duplex
PCI-Express (1-lane)	50	Full Duplex

Table 5.4: Sample rates for various host interfaces [9].

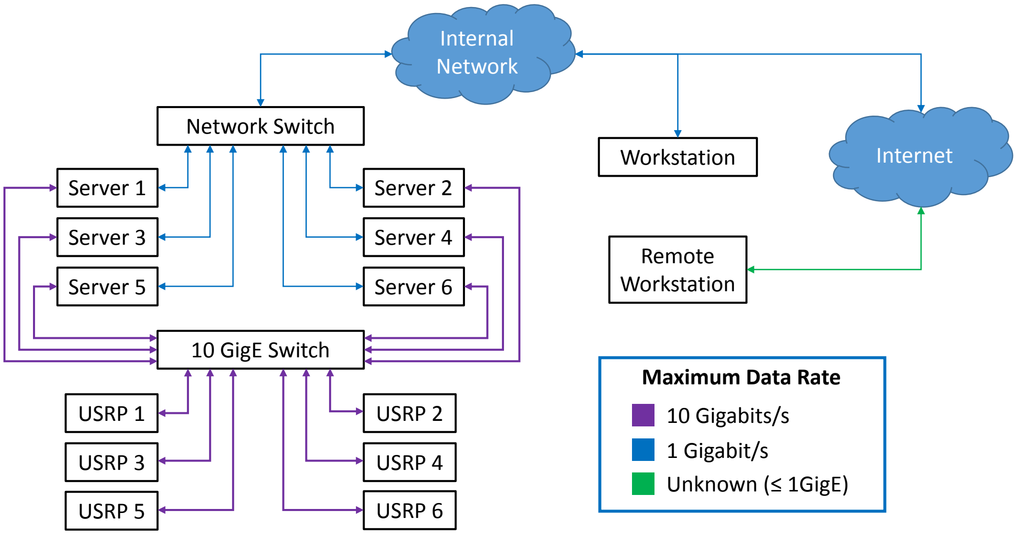


Figure 5.27: Network Layout with maximum throughput.

Care was taken to keep network throughput at 10GigE between the USRPs and the servers controlling them. For the local and remote workstations, the demands on throughput can be relaxed significantly since the samples data never travels over those connections.

Another interesting note is that the storage capabilities of the servers will place limitations on the system. For example, the file size for a single RF channel sampling at 200 MSps is

$$size = tR_b = 32tR_s = \frac{32bits}{S}(30s)(200MSps) = 192,000Mb = 24,000MB = 24GB$$

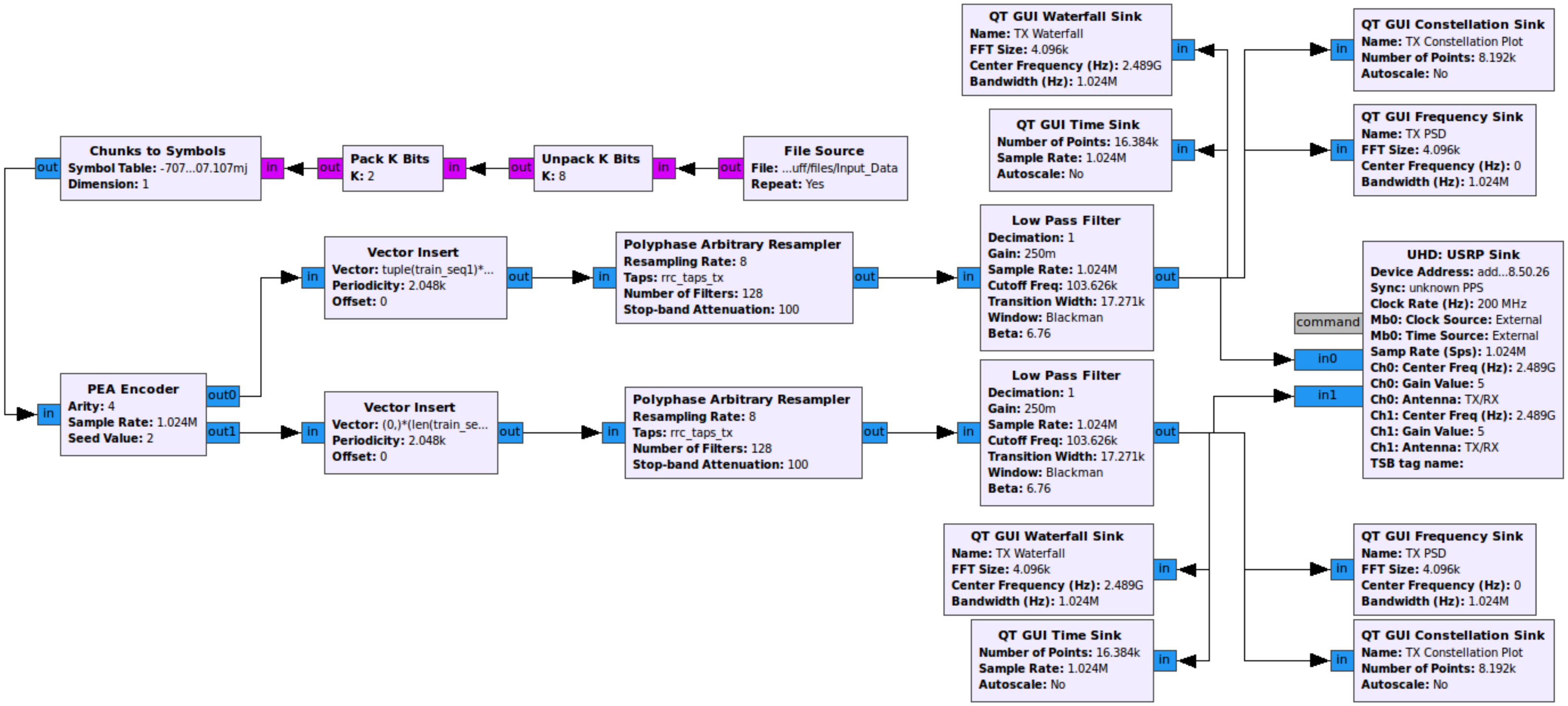
A 30 second snapshot for a single channel takes up 24 GB of disk space! In fact, due to the RAID setup on the servers (RAID 10) the data is stored twice, doubling the required space to 48 GB. Four 800 GB drives are installed on the server, so these kinds of measurements should not pose a problem if they are under a few minutes in length.

Chapter 6

PEAC Design

6.1 Transmitter

The design of the PEAC transmitter in Figure 6.1 is broken up into the aspects of data management, modulation, space-time coding, pulse shaping, and interfacing with the RF front-end device.



QT GUI Waterfall Sink
Name: TX Waterfall
FFT Size: 4.096k
Center Frequency (Hz): 2.489G
Bandwidth (Hz): 1.024M

QT GUI Constellation Sink
Name: TX Constellation Plot
Number of Points: 8.192k
Autoscale: No

Chunks to Symbols
Symbol Table: -707...07.107mj
Dimension: 1

Pack K Bits
K: 2

Unpack K Bits
K: 8

File Source
File: ...uff/files/Input_Data
Repeat: Yes

QT GUI Time Sink
Number of Points: 16.384k
Sample Rate: 1.024M
Autoscale: No

QT GUI Frequency Sink
Name: TX PSD
FFT Size: 4.096k
Center Frequency (Hz): 0
Bandwidth (Hz): 1.024M

Low Pass Filter
Decimation: 1
Gain: 250m
Sample Rate: 1.024M
Cutoff Freq: 103.626k
Transition Width: 17.271k
Window: Blackman
Beta: 6.76

Vector Insert
Vector: tuple(train_seq1)*...
Periodicity: 2.048k
Offset: 0

Polyphase Arbitrary Resampler
Resampling Rate: 8
Taps: rrc_taps_tx
Number of Filters: 128
Stop-band Attenuation: 100

UHD: USRP Sink
Device Address: add...8.50.26
Sync: unknown PPS
Clock Rate (Hz): 200 MHz
Mb0: Clock Source: External
Mb0: Time Source: External
Samp Rate (Sps): 1.024M
Ch0: Center Freq (Hz): 2.489G
Ch0: Gain Value: 5
Ch0: Antenna: TX/RX
Ch1: Center Freq (Hz): 2.489G
Ch1: Gain Value: 5
Ch1: Antenna: TX/RX
TSB tag name:

PEA Encoder
Arity: 4
Sample Rate: 1.024M
Seed Value: 2

Vector Insert
Vector: (0,)*(len(train_se...
Periodicity: 2.048k
Offset: 0

Polyphase Arbitrary Resampler
Resampling Rate: 8
Taps: rrc_taps_tx
Number of Filters: 128
Stop-band Attenuation: 100

Low Pass Filter
Decimation: 1
Gain: 250m
Sample Rate: 1.024M
Cutoff Freq: 103.626k
Transition Width: 17.271k
Window: Blackman
Beta: 6.76

QT GUI Waterfall Sink
Name: TX Waterfall
FFT Size: 4.096k
Center Frequency (Hz): 2.489G
Bandwidth (Hz): 1.024M

QT GUI Frequency Sink
Name: TX PSD
FFT Size: 4.096k
Center Frequency (Hz): 0
Bandwidth (Hz): 1.024M

QT GUI Time Sink
Number of Points: 16.384k
Sample Rate: 1.024M
Autoscale: No

QT GUI Constellation Sink
Name: TX Constellation Plot
Number of Points: 8.192k
Autoscale: No

Figure 6.1: PEAC Transmitter.

6.1.1 Data Management and Digital Modulation

The data management and digital modulation portions of the PEAC transmitter are shown in Figure 6.2. QPSK-modulated symbols are generated in a similar way to the QPSK OTA flowgraph described in chapter 5. The major difference in the PEAC flowgraph is how the header and payload data are handled. Since the STBC is only applied to the payload, the encoding is done before applying a header. The header is also applied in a different manner - with the `Vector Insert` block shown in Figure 6.4.

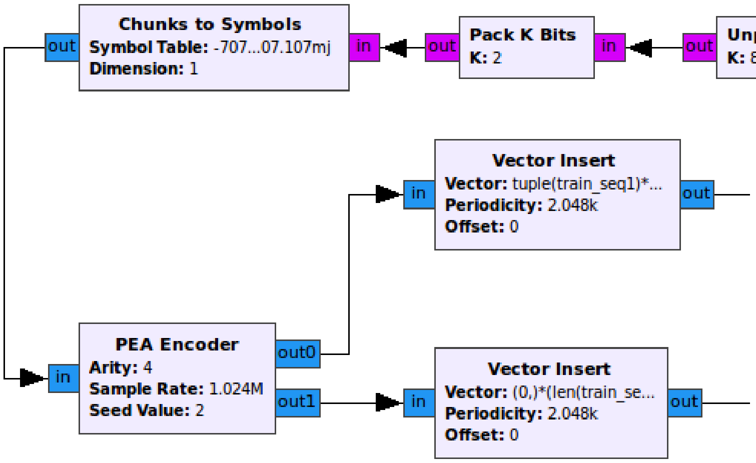


Figure 6.2: Data Management in the PEAC Transmitter.

The header design is displayed in Figure 6.3. Unique Words (UWs) are used to sound the channel for each antenna. Therefore, each of the UWs are placed in a Time Domain Multiple Access (TDMA) structure to avoid interference. The details of how the header is used to perform frame synchronization and channel estimation are discussed in the description of the receiver.

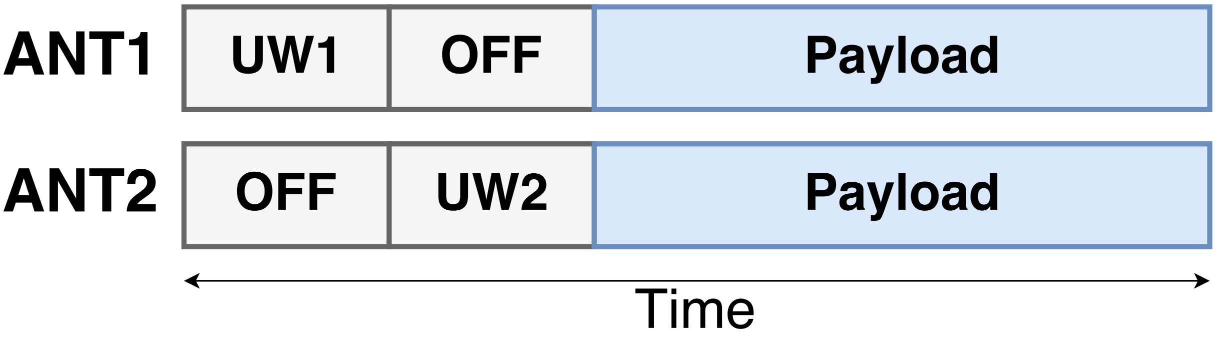
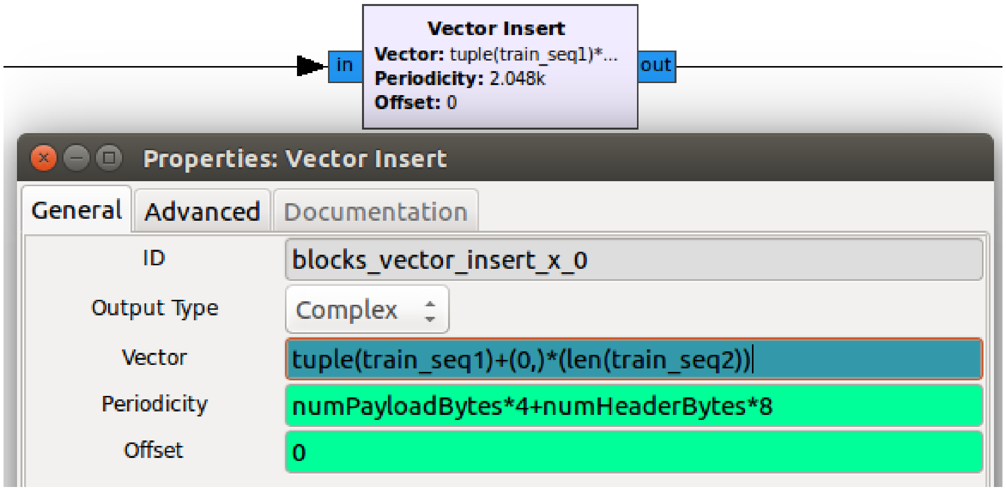


Figure 6.3: Header Design for the PEAC System.

Vector Insert

Figure 6.4 shows the `Vector Insert` block and the parameters used for antenna 1 in the PEAC transmitter flowgraph. The Vector parameter specifies the vector inserted into the stream. The periodicity and offset parameters specify how often and in what position the vector is inserted, respectively.

In order to packetize a stream of payload data with the `Vector Insert` block, the header data is specified in the vector field, the total length of the packet is put in the Periodicity field, and the offset is set to 0. This way, the header data is inserted every packet length before the payload.



Vector Insert
Vector: tuple(train_seq1)*...
Periodicity: 2.048k
Offset: 0

Properties: Vector Insert

ID	blocks_vector_insert_x_0
Output Type	Complex
Vector	tuple(train_seq1)+(0,)*(len(train_seq2))
Periodicity	numPayloadBytes*4+numHeaderBytes*8
Offset	0

Figure 6.4: `Vector Insert` Block and its relevant parameters.

6.1.2 Space-Time Encoding

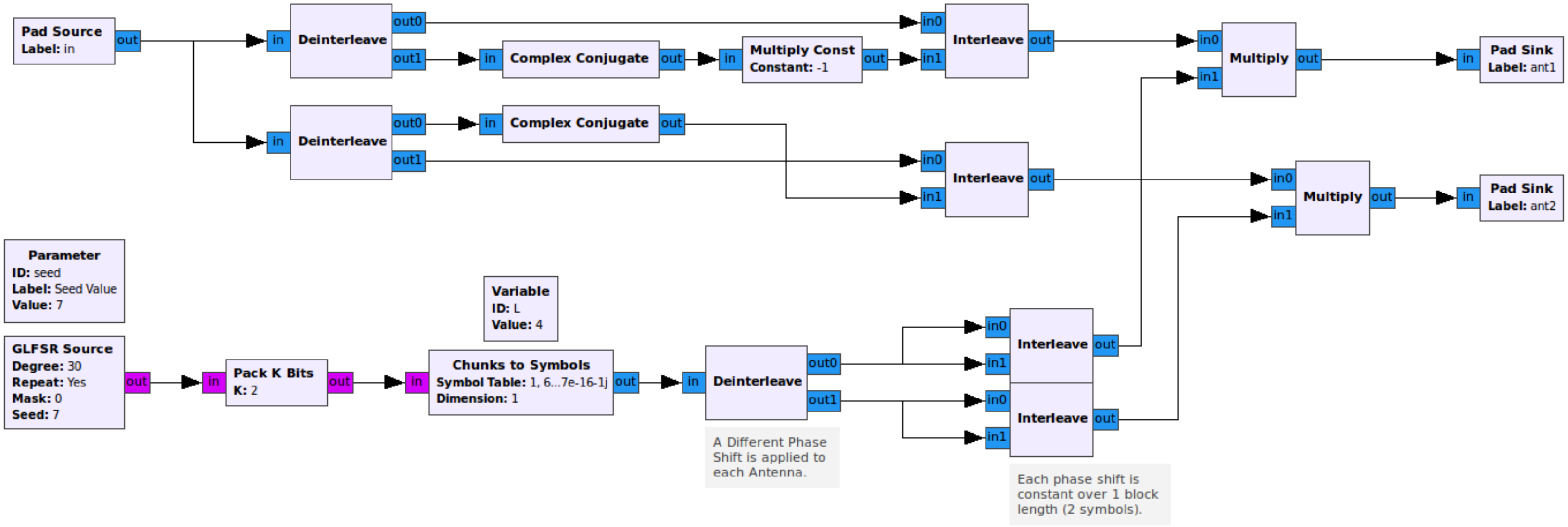
The PEAC space-time encoder is derived from the alamouti encoder in [40]. The flowgraph for the PEAC space-time encoder is shown in Figure 6.5.

For reference, the PEAC scheme is coded

$$\mathbf{X_A} = \begin{bmatrix} s_1 e^{j\theta_1} & -s_2^* e^{j\theta_1} \\ s_2 e^{j\theta_2} & s_1^* e^{j\theta_2} \end{bmatrix}$$

where the rows of X represent different transmit antennas and the columns correspond to time slots.

The upper half of Figure 6.5 implements alamouti coding by first deinterleaving an input stream into s_1 and s_2 streams. After the coding is applied to each stream, they are interleaved back together. The pseudo-random phase shifts are generated in the bottom half of Figure 6.5 and are applied to each transmit antenna.



A Different Phase Shift is applied to each Antenna.

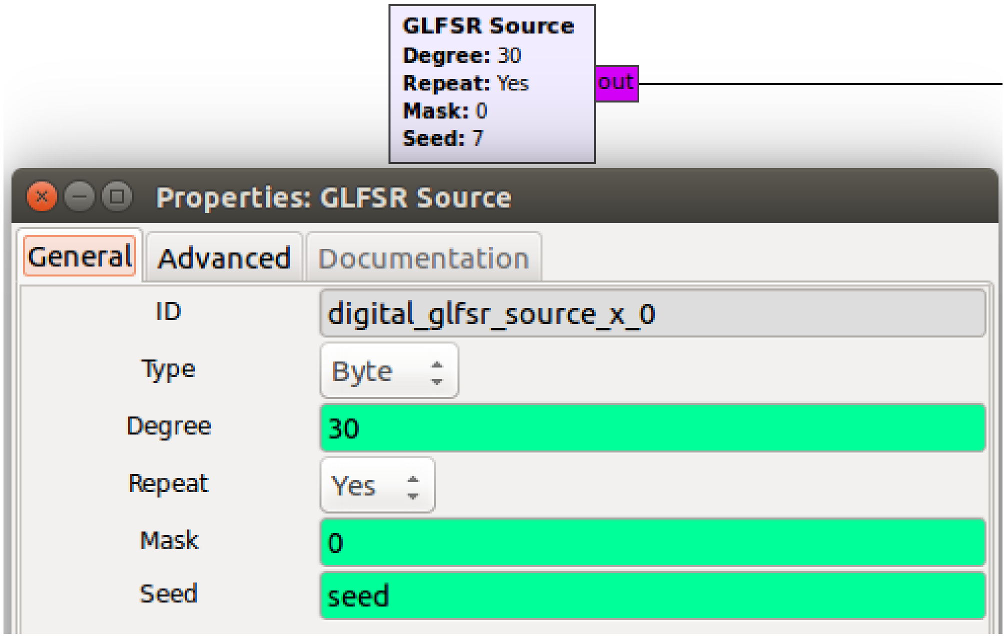
Each phase shift is constant over 1 block length (2 symbols).

Figure 6.5: PEAC Space-Time Encoder.

GLFSR Source

Figure 6.6 shows the Galois Linear Feedback Shift Register (GLFSR) Source and its relevant parameters. The block generates numbers from a psuedo-random sequence specified by the degree, mask, and seed parameters. The degree sets the length of the sequence, for the PEAC system this length is $2^{30} \approx 10^{9}$. The mask determines the generator polynomials for the sequence which can completely change the resulting sequence. The seed determines the initial fill of the registers used to generate the sequence, and will offset the sequence by the specified value.

In the case of the PEAC system, a very long pseudo-random sequence is generated with the `GLFSR` block and the seed is used as a key. This means that Eve will know the sequence being used, but will not know the offset of the sequence. This is similar to how GPS Fine Acquisition (FA) codes are implemented.



GLFSR Source
Degree: 30
Repeat: Yes
Mask: 0
Seed: 7

ID: digital_glfsr_source_x_0
Type: Byte
Degree: 30
Repeat: Yes
Mask: 0
Seed: seed

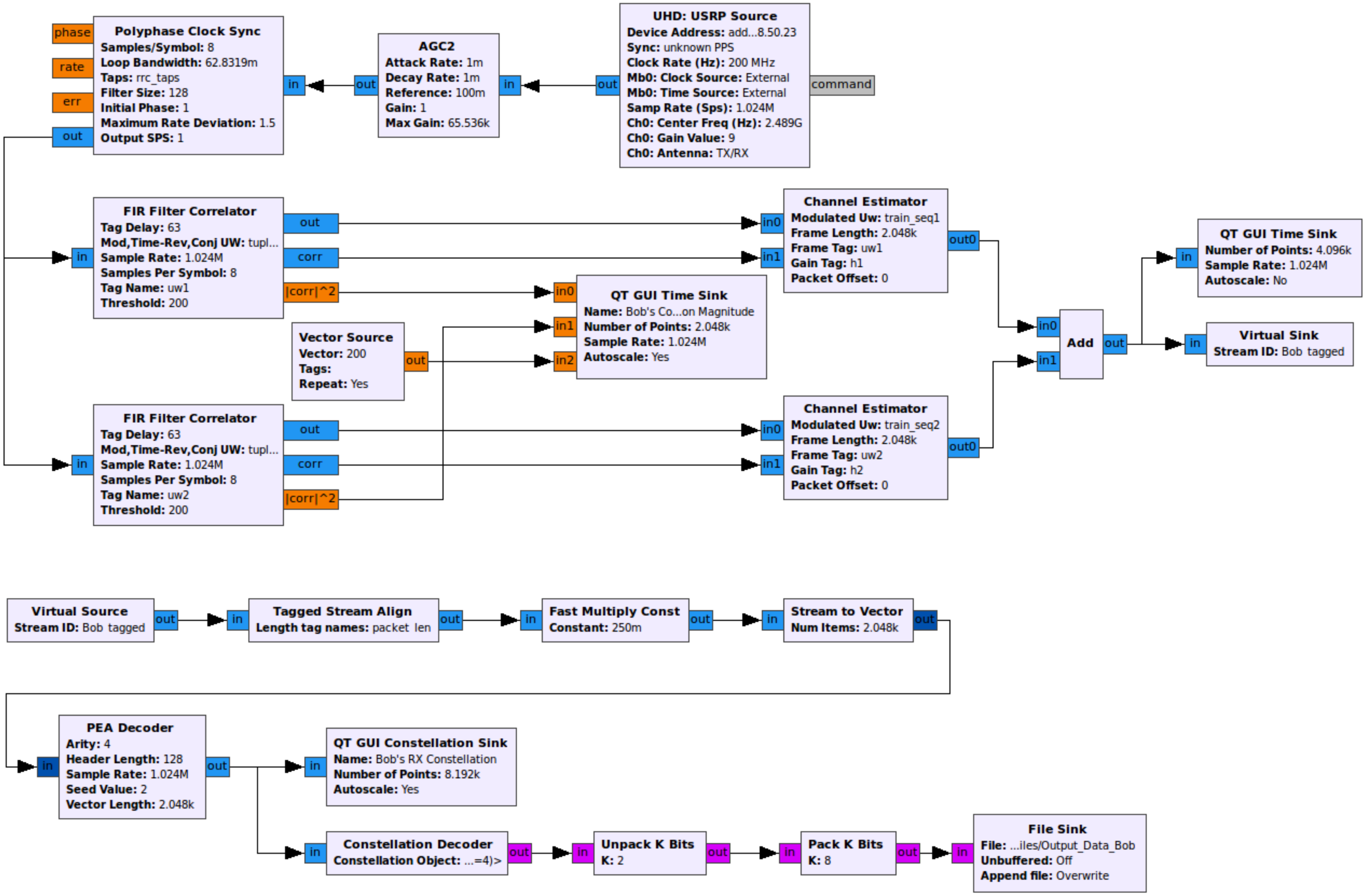
Figure 6.6: GLFSR Source Block and its relevant parameters.

6.1.3 Pulse Shaping and RF Front-End Interface

The QPSK symbols are then up-sampled in the `Polyphase Arbitrary Resampler` block which also perform pulse shaping with a RRC filter.

6.2 Receiver

Bob's receiver is designed based on the QPSK receiver, but adds in the additional components of space-time coding, packet synchronization, and channel estimation. Additionally the PEAC receiver handles gain control and recovery loops.



Polyphase Clock Sync
Samples/Symbol: 8
Loop Bandwidth: 62.8319m
Taps: rrc_taps
Filter Size: 128
Initial Phase: 1
Maximum Rate Deviation: 1.5
Output SPS: 1

AGC2
Attack Rate: 1m
Decay Rate: 1m
Reference: 100m
Gain: 1
Max Gain: 65.536k

UHD: USRP Source
Device Address: add...8.50.23
Sync: unknown PPS
Clock Rate (Hz): 200 MHz
Mb0: Clock Source: External
Mb0: Time Source: External
Samp Rate (Sps): 1.024M
Ch0: Center Freq (Hz): 2.489G
Ch0: Gain Value: 9
Ch0: Antenna: TX/RX

FIR Filter Correlator
Tag Delay: 63
Mod,Time-Rev,Conj UW: tupl...
Sample Rate: 1.024M
Samples Per Symbol: 8
Tag Name: uw1
Threshold: 200

Channel Estimator
Modulated Uw: train_seq1
Frame Length: 2.048k
Frame Tag: uw1
Gain Tag: h1
Packet Offset: 0

QT GUI Time Sink
Number of Points: 4.096k
Sample Rate: 1.024M
Autoscale: No

QT GUI Time Sink
Name: Bob's Co...on Magnitude
Number of Points: 2.048k
Sample Rate: 1.024M
Autoscale: Yes

Vector Source
Vector: 200
Tags:
Repeat: Yes

Add

Virtual Sink
Stream ID: Bob tagged

FIR Filter Correlator
Tag Delay: 63
Mod,Time-Rev,Conj UW: tupl...
Sample Rate: 1.024M
Samples Per Symbol: 8
Tag Name: uw2
Threshold: 200

Channel Estimator
Modulated Uw: train_seq2
Frame Length: 2.048k
Frame Tag: uw2
Gain Tag: h2
Packet Offset: 0

Virtual Source
Stream ID: Bob tagged

Tagged Stream Align
Length tag names: packet_len

Fast Multiply Const
Constant: 250m

Stream to Vector
Num Items: 2.048k

PEA Decoder
Arity: 4
Header Length: 128
Sample Rate: 1.024M
Seed Value: 2
Vector Length: 2.048k

QT GUI Constellation Sink
Name: Bob's RX Constellation
Number of Points: 8.192k
Autoscale: Yes

Constellation Decoder
Constellation Object: ...=4)>

Unpack K Bits
K: 2

Pack K Bits
K: 8

File Sink
File: ...iles/Output_Data_Bob
Unbuffered: Off
Append file: Overwrite

Figure 6.7: Bob's PEAC Receiver.

6.2.1 Gain Control and Recovery Loops

The USRP functioning as the receiver is tuned with the `UHD: USRP Source` block which provides complex baseband samples from the USRP.

First, gain control is performed with the `AGC2` block. The next step in the receiver chain is to eliminate any timing offset between the clock at receiver and transmitter. The timing recovery and matched filtering are implemented with a `Polyphase Clock Sync` block.

Alamouti coding with QPSK does not lend itself to easy blind recovery of the carrier used by the transmitter. Since both transmit elements will send independent QPSK symbols, the received constellation is not QPSK, but rather a 3×3 grid of points formed by the combination of all possible QPSK symbols. This constellation is shown in Figure 6.8. The costas loops used in GNU Radio either in the `Costas Loop` block or more generally in the `Constellation Receiver` block are not designed to perform frequency recovery of constellations without a constant modulus such as QAM or the constellation in Figure 6.8. Instead, the transmitter and receiver are connected to a common clock and timing

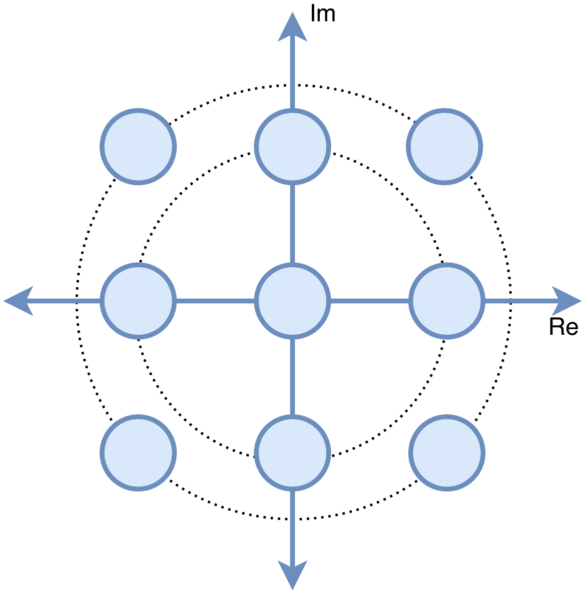


Figure 6.8: Received Constellation for Alamouti Coding with QPSK.

source which eliminates the need to recover the carrier frequency at the receiver.

6.2.2 Packet Synchronization

To synchronize with a packet, the receiver correlates with the unique word using a `Decimating FIR Filter` block. The discrete complex cross-correlation of two sequences $p[n]$ and $u[n]$ of length N is defined as

$$R(p,u) = \sum_{m=0}^{N-1} p[m]u^*[m-n].$$

A FIR filter of order $N-1$ performs a convolution operation of the input $x[n]$ with the filter taps $h[n]$ to produce the output

$$y[n] = \sum_{m=0}^{N-1} x[m]h[n-m].$$

By substituting the taps

$$h_{corr}[n] = u^*[-n]$$

into the FIR filter, the output becomes

$$y_{corr}[n] = \sum_{m=0}^{N-1} x[m]u^*[m-n] = R(x,u).$$

Therefore, to implement the cross-correlation of the input with the unique word using a FIR filter, the filter taps are set to the time-reversed and conjugated unique word. The hierarchical block implementing the `FIR Filter Correlator` is shown in Figure 6.9.

The unique words shown in Figure 6.3 are chosen to be 63-bit Maximum-Length Sequences (MLS) that are zero padded to 64 bits. The MLSs were chosen to have ideal autocorrelation and cross-correlation properties and the zero-padding was done to make handling the sequence easier in GNU Radio.

FIR Filter Correlator

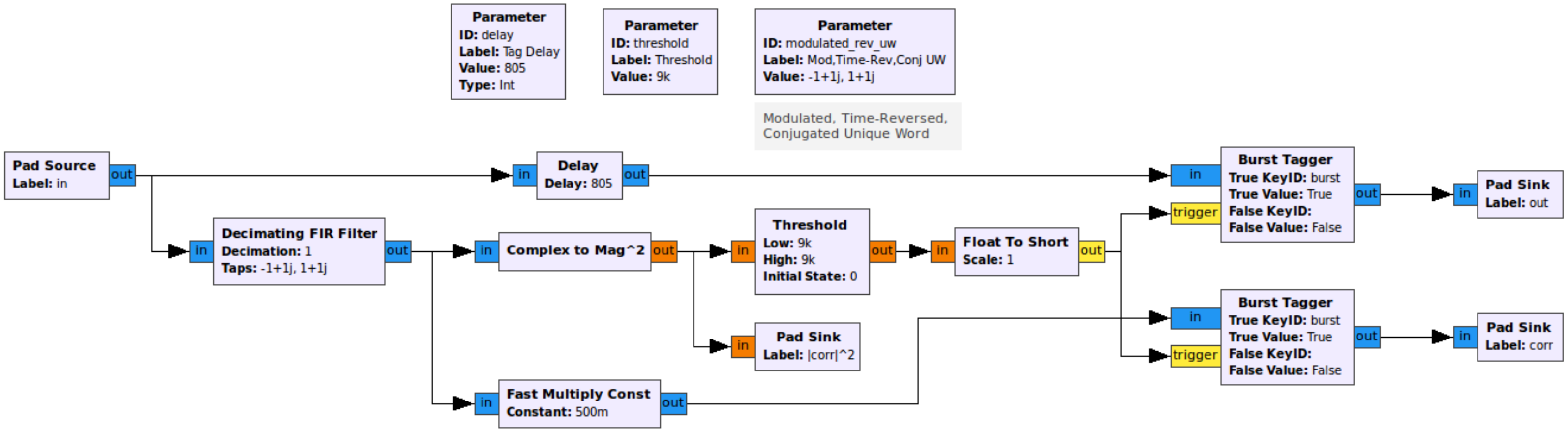


Figure 6.9: `FIR Filter Correlator` hierarchical block.

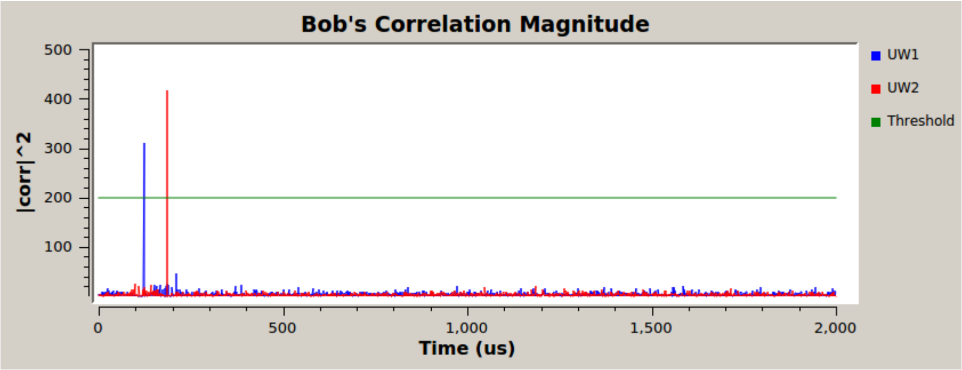


Figure 6.10: An example of the magnitude-squared correlation filter outputs for each unique word and threshold level.

An example of the magnitude-squared values of the output for the two correlation filters in Figure 6.7 is shown in Figure 6.10. This graph corresponds to output seen with the `QT GUI Time Sink` in Figure 6.7. The magnitude-squared correlation value is used with the `Threshold` block to trigger a `Burst Tagger` which applies a stream tag to the input data when a correlation spike occurs. Since the correlation filter introduces a delay into the stream, a `Delay` block matches the input branch to the correlation branch to appropriately align the stream tag.

6.2.3 Channel Estimation

The output of the correlation filter can also be used to perform an estimate of the channel gains. Consider the cross-correlation of a unique word $\mathbf{w}$ of length N and amplitude 1 with the same unique word that experiences a complex Rayleigh flat fading gain h

$$R(h\mathbf{w}[n], \mathbf{w}[n]) = \sum_{m=0}^{N-1} h\mathbf{w}[m]\mathbf{w}^*[m-n].$$

The peak value occurs when the unique words overlap at $n = 0$

$$R_{peak} = \sum_{m=0}^{N-1} h\mathbf{w}[m]\mathbf{w}^*[m] = Nh.$$

Therefore, the channel gain h can be estimated from the value of the correlation peak and the length of the unique word.

Channel Estimator Block

An example of the complex values of the output for a single correlation filter in Figure 6.7 is shown in Figure 6.11. This output was derived from a simulated flat-fading channel with Rayleigh fading gain $h = 0.2 - 0.3j$. Notice that by measuring the peak real and imaginary values, which occur simultaneously, the channel gain can be accurately measured. This is because the `FIR Filter Correlator` block already provided the necessary scaling by the unique word length. The stream tag "uw1" has been applied at the peak value by the `FIR Filter Correlator` as described previously. The `Channel Estimator` block shown in Figure 6.12 reads the value at this peak to determine the channel gain and adds the estimates as stream tags to each unique word in the received signal.

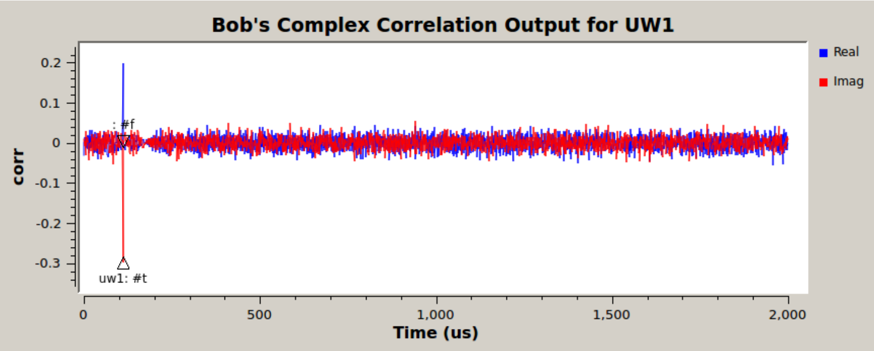
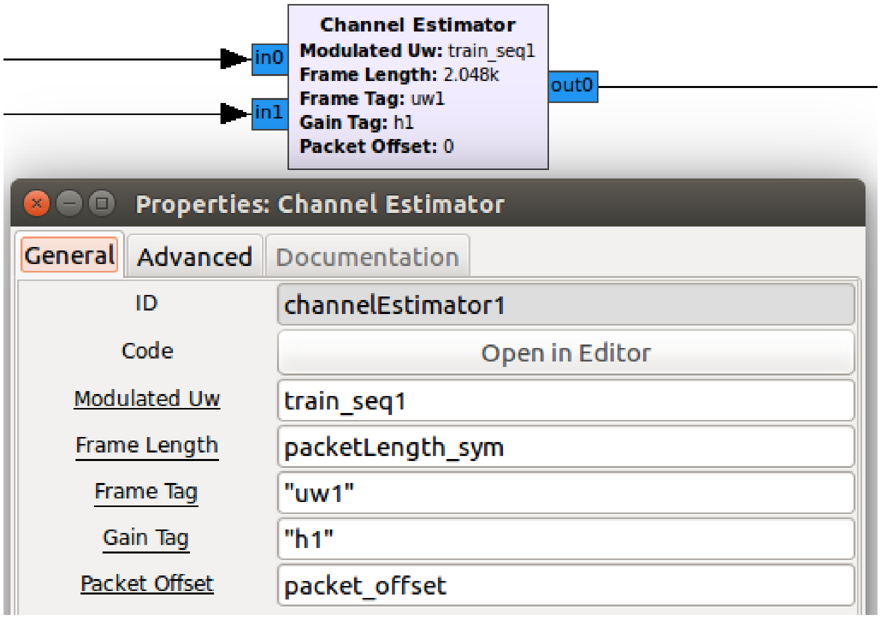


Figure 6.11: An example of the complex correlation filter output for one of the unique words with Rayleigh fading channel gain $h = 0.2 - 0.3j$.



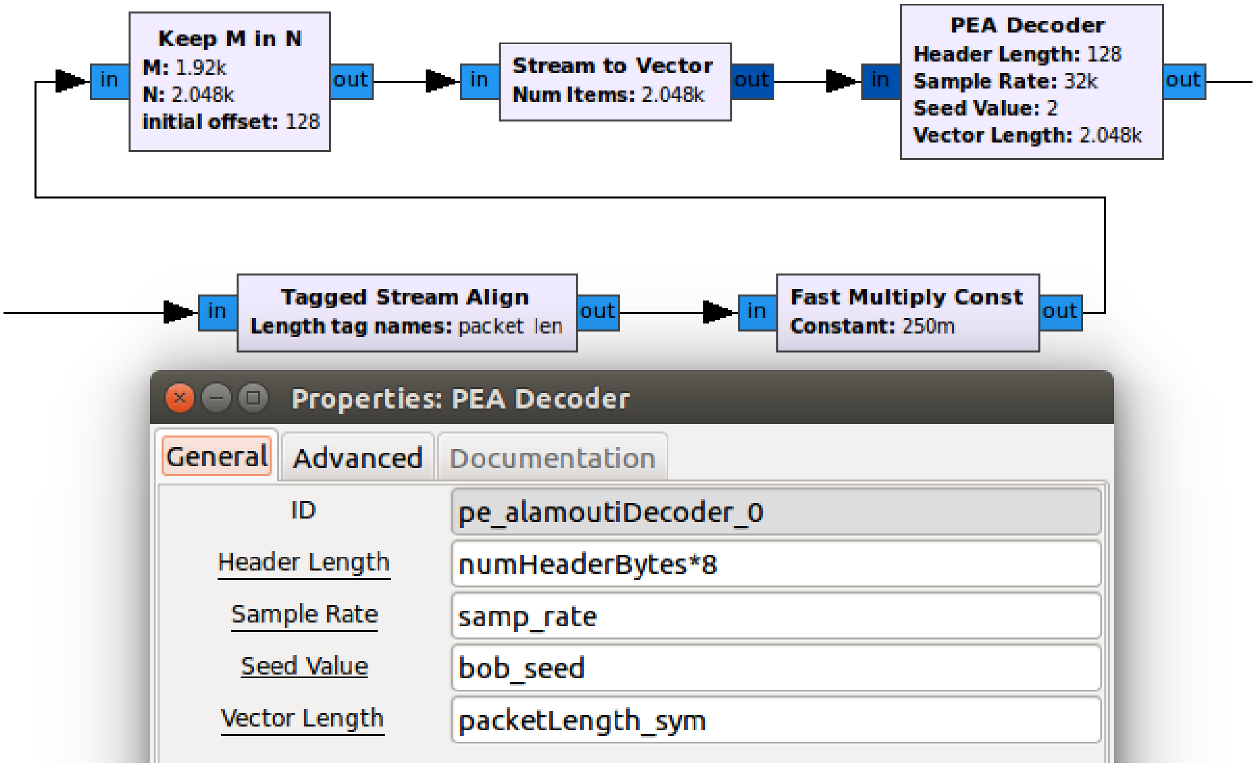
Channel Estimator
Modulated Uw: train_seq1
Frame Length: 2.048k
Frame Tag: uw1
Gain Tag: h1
Packet Offset: 0

ID	channelEstimator1
Code	Open in Editor
Modulated Uw	train_seq1
Frame Length	packetLength_sym
Frame Tag	"uw1"
Gain Tag	"h1"
Packet Offset	packet_offset

Figure 6.12: The `Channel Estimator` block with its properties.

6.2.4 Space-Time Decoding

The `PEA Decoder` block implements the Alamouti equalization while undoing the pseudo-random phase shifts applied at the transmitter. Prior to being fed into the `PEA Decoder` block, the stream is aligned based on the UW1 tag added by the `FIR Filter Correlator` block and then each packet is converted into a vector with the `Stream to Vector` block. These blocks, along with the properties for the `PEA Decoder`, are shown in Figure 6.13.



Keep M in N
M: 1.92k
N: 2.048k
initial offset: 128

Stream to Vector
Num Items: 2.048k

PEA Decoder
Header Length: 128
Sample Rate: 32k
Seed Value: 2
Vector Length: 2.048k

Tagged Stream Align
Length tag names: packet_len

Fast Multiply Const
Constant: 250m

ID: pe_alamoutiDecoder_0
Header Length: numHeaderBytes*8
Sample Rate: samp_rate
Seed Value: bob_seed
Vector Length: packetLength_sym

Figure 6.13: The `PEA Decoder` block with its properties along with the preprocessing blocks used to convert the stream to a packet vector.

In the decoder, the stream tags for each channel gain are used to generate the channel estimates for each packet. A `GLFSR` block with a corresponding seed value to the transmitter applies inverted phase shifts to undo the phase offset in the channel estimate.

6.3 Eve's Receiver

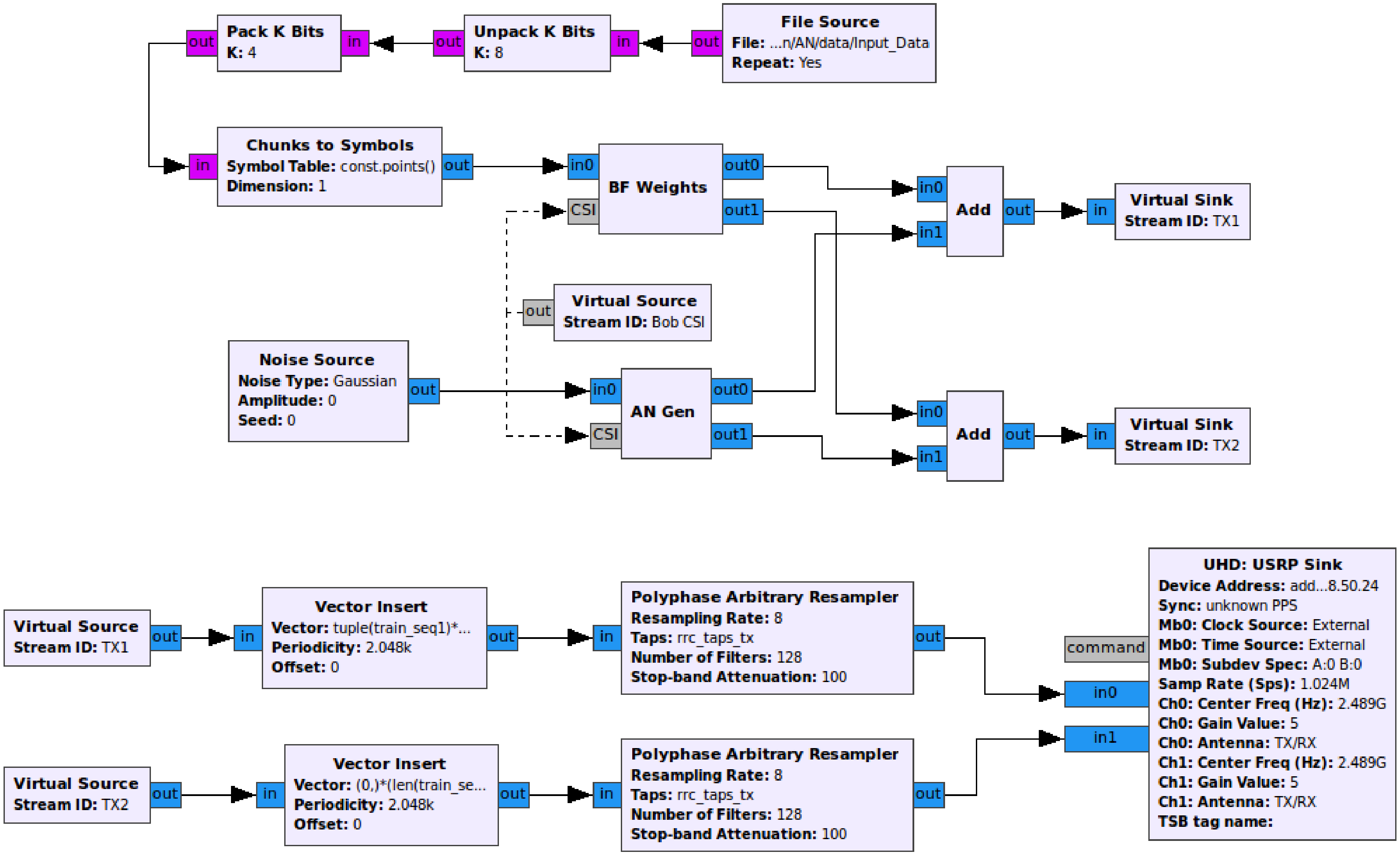
Eve's receiver is designed to be equivalent to Bob's, with the only difference being that Eve won't have the correct seed value for the `PEA Decoder` block.

Chapter 7

Artificial Noise Design

7.1 Transmitter

The AN transmitter's data management, header design, modulation, and pulse shaping are handled identically to the PEAC transmitter. The major differences between the AN and PEAC transmitters are the change from an Alamouti STBC in the PEAC system to a transmit beamformer in the AN system and the addition of an AN generator to the AN transmitter. Figure 7.1 displays the AN transmitter



File Source
File: ...n/AN/data/Input_Data
Repeat: Yes

Unpack K Bits
K: 8

Pack K Bits
K: 4

Chunks to Symbols
Symbol Table: const.points()
Dimension: 1

BF Weights

Virtual Source
Stream ID: Bob CSI

Noise Source
Noise Type: Gaussian
Amplitude: 0
Seed: 0

AN Gen

Add

Virtual Sink
Stream ID: TX1

Add

Virtual Sink
Stream ID: TX2

Virtual Source
Stream ID: TX1

Vector Insert
Vector: tuple(train_seq1)*...
Periodicity: 2.048k
Offset: 0

Polyphase Arbitrary Resampler
Resampling Rate: 8
Taps: rrc_taps_tx
Number of Filters: 128
Stop-band Attenuation: 100

Virtual Source
Stream ID: TX2

Vector Insert
Vector: (0,)*(len(train_se...
Periodicity: 2.048k
Offset: 0

Polyphase Arbitrary Resampler
Resampling Rate: 8
Taps: rrc_taps_tx
Number of Filters: 128
Stop-band Attenuation: 100

UHD: USRP Sink
Device Address: add...8.50.24
Sync: unknown PPS
Mb0: Clock Source: External
Mb0: Time Source: External
Mb0: Subdev Spec: A:0 B:0
Samp Rate (Sps): 1.024M
Ch0: Center Freq (Hz): 2.489G
Ch0: Gain Value: 5
Ch0: Antenna: TX/RX
Ch1: Center Freq (Hz): 2.489G
Ch1: Gain Value: 5
Ch1: Antenna: TX/RX
TSB tag name:

Figure 7.1: AN Transmitter.

7.1.1 Transmit Beamforming

The transmitter makes use of CSI feedback from Bob to perform transmit beamforming with the `BF Weights` block. It does this in the manner described in section 4.1.2.

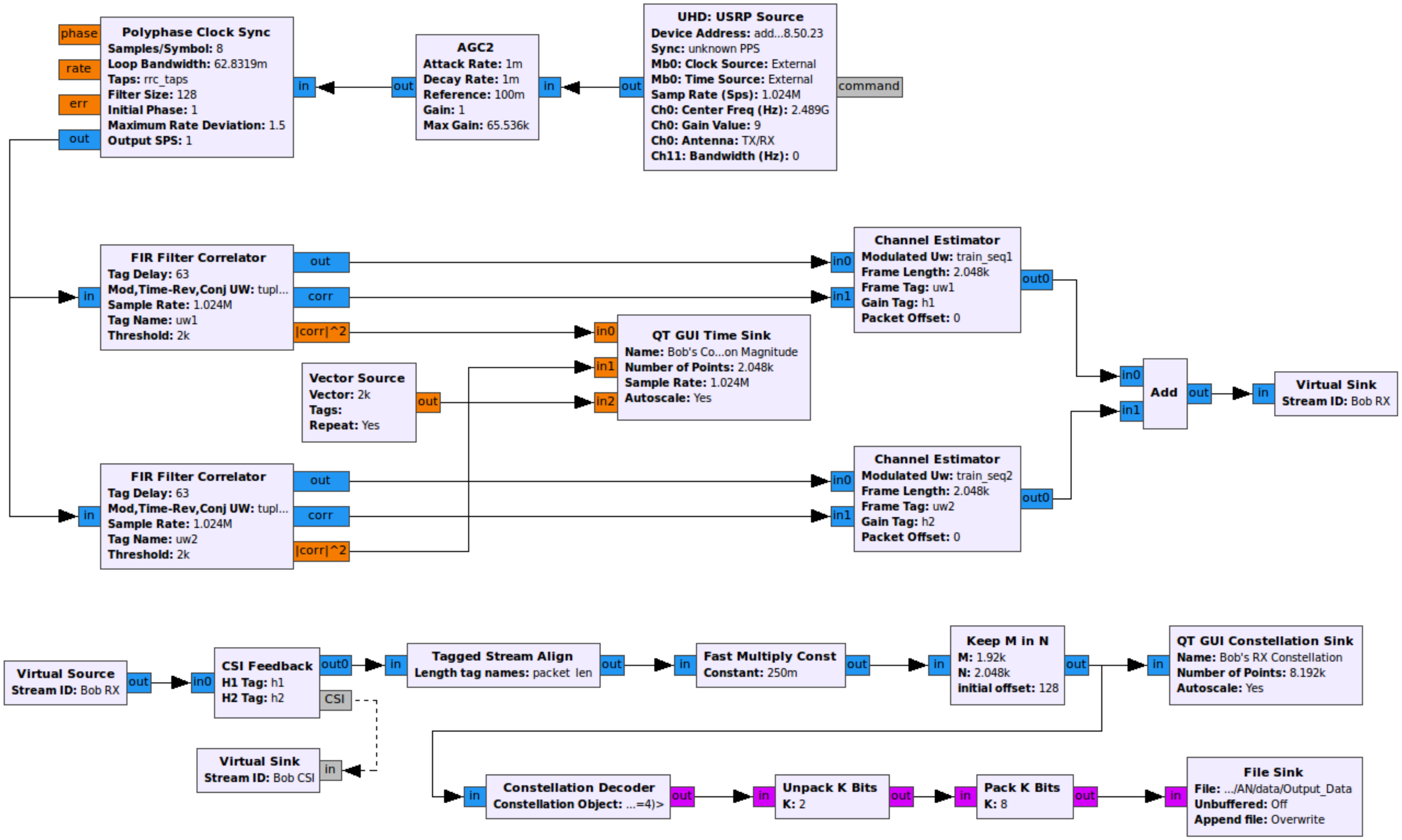
7.1.2 Artificial Noise Generation

For a 2×1 system, Alice generates AN such that $\mathbf{h}_k \mathbf{w}_k = 0$. $\mathbf{w}_k$ is generated from $\mathbf{w}_k = \mathbf{z}_k v$ where v is a Gaussian distributed complex scaler and $\mathbf{z}_k$ is a unit orthonormal basis vector for the nullspace of $\mathbf{h}_k$. The general case is described in Section 4.1.

The AN is generated using a Gaussian `Noise Source` block to generate v multiplied by $\mathbf{z}$ which is generated from Bob's CSI feedback. The AN generation is implemented by the `AN Gen` block shown in Figure 7.1. The top and bottom entries in $\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$ are applied to the first and second antennas, respectively.

7.2 Bob's Receiver

The AN receiver's recovery loops, packet synchronization, and channel estimation are implemented identically to the PEAC receiver. The major difference between the AN and PEAC receivers is the addition of CSI feedback in the former. Figure 7.2 displays Bob's AN receiver.



Polyphase Clock Sync
Samples/Symbol: 8
Loop Bandwidth: 62.8319m
Taps: rrc_taps
Filter Size: 128
Initial Phase: 1
Maximum Rate Deviation: 1.5
Output SPS: 1

AGC2
Attack Rate: 1m
Decay Rate: 1m
Reference: 100m
Gain: 1
Max Gain: 65.536k

UHD: USRP Source
Device Address: add...8.50.23
Sync: unknown PPS
Mb0: Clock Source: External
Mb0: Time Source: External
Samp Rate (Sps): 1.024M
Ch0: Center Freq (Hz): 2.489G
Ch0: Gain Value: 9
Ch0: Antenna: TX/RX
Ch11: Bandwidth (Hz): 0

FIR Filter Correlator
Tag Delay: 63
Mod,Time-Rev,Conj UW: tupl...
Sample Rate: 1.024M
Tag Name: uw1
Threshold: 2k

Channel Estimator
Modulated Uw: train_seq1
Frame Length: 2.048k
Frame Tag: uw1
Gain Tag: h1
Packet Offset: 0

QT GUI Time Sink
Name: Bob's Co...on Magnitude
Number of Points: 2.048k
Sample Rate: 1.024M
Autoscale: Yes

Vector Source
Vector: 2k
Tags:
Repeat: Yes

Add

Virtual Sink
Stream ID: Bob RX

FIR Filter Correlator
Tag Delay: 63
Mod,Time-Rev,Conj UW: tupl...
Sample Rate: 1.024M
Tag Name: uw2
Threshold: 2k

Channel Estimator
Modulated Uw: train_seq2
Frame Length: 2.048k
Frame Tag: uw2
Gain Tag: h2
Packet Offset: 0

Virtual Source
Stream ID: Bob RX

CSI Feedback
H1 Tag: h1
H2 Tag: h2

Tagged Stream Align
Length tag names: packet_len

Fast Multiply Const
Constant: 250m

Keep M in N
M: 1.92k
N: 2.048k
initial offset: 128

QT GUI Constellation Sink
Name: Bob's RX Constellation
Number of Points: 8.192k
Autoscale: Yes

Virtual Sink
Stream ID: Bob CSI

Constellation Decoder
Constellation Object: ...=4)>

Unpack K Bits
K: 2

Pack K Bits
K: 8

File Sink
File: .../AN/data/Output_Data
Unbuffered: Off
Append file: Overwrite

Figure 7.2: Bob's AN Receiver.

In the AN technique, CSI is required at the transmitter to generate noise in the nullspace of Bob's channel and to perform transmit beamforming. To facilitate this in a test environment, the CSI is sent to the transmitter using GNU Radio's asynchronous messaging protocol which updates the `AN Gen` and `BF Weights` blocks.

7.3 Eve's Receiver

Eve's receiver functions identically to Bob's except that the CSI feedback mechanism is not implemented.

Chapter 8

Experimental Results

The goal of the experiments performed on the AN system was to produce an experimental BER vs. SNR curve to compare to the theoretical results generated in chapter 4. This chapter describes the methodology and the results of these experiments.

The testbed was configured such that the transmitter USRP and receiver USRP were in a Non-Line-of-Sight (NLOS) channel, indoors in a laboratory environment with obstacles to provide multipath scattering. A summary of the parameters used for the test is shown in Table 8.1.

Parameter	**Value**
Sample Rate	1 MSps
Transmit Frequency	2.489 GHz
GLFSR Order	24
Number of Antennas (Alice by Bob by Eve)	2x1x1

Table 8.1: Parameters used during experiment.

The data sequence used for source data in the test was generated using a GLFSR. A GLFSR of order n produces a repeating psuedo-random sequence of length $2^n - 1$ which will contain every permutation of n bits. This is a great property for a BER test since the channel may effect different sequences of bits in a unique way. By using every permutation of bits, a more accurate average of the system's BER for the given channel is attained.

To vary the SNR at the receiver, the gain at the transmitter was incremented from 0 dB to 20 dB in intervals of 0.5 dB for each run. The GRC Flowgraph used to run the experiments is a combination of Figure 7.1 and Figure 7.2 with added `File Sinks` to collect symbols and bits at verious stages of the processing.

Two MATLAB scripts are used to measure the SNR and the BER of each received signal. The first MATLAB script processes the received complex symbols from a single run, and measures the Error Vector Magnitude (EVM) of each constellation point. The script calculates the EVM by comparing the received symbols to an expected constellation. The EVM is then determined by measuring the magnitude of the difference between the expected and received constellation points. Finally, the SNR is computed through the inversely proportional relationship of EVM and SNR shown in [41],

$$EVM_{RMS} = \sqrt{1/SNR} = \sqrt{N_0/E_s}$$

where $N_0/2$ is the noise power spectral density and E_s is the symbol energy. This relationship is only completely accurate in a data-aided communication system, where the source data can be compared at the receiver to determine EVM. In non data-aided systems, there will be an error in the estimation of EVM at low values of SNR.

The second MATLAB script calculates the BER by aligning the received bits with the source bits properly and comparing them to each other. The measurements of both scripts were repeated over 10 runs and the results were averaged to produced the final experimental BER vs. SNR curves.

This test was run in two different scenarios. The first scenario is a baseline where no AN is transmitted, the BER vs. SNR curve with no AN is shown in Figure 8.1. In the second scenario, the transmitter uses AN to mask the transmission with a power ratio of $\alpha = 0.5$. The BER vs. SNR curve with AN is shown in Figure 8.2.

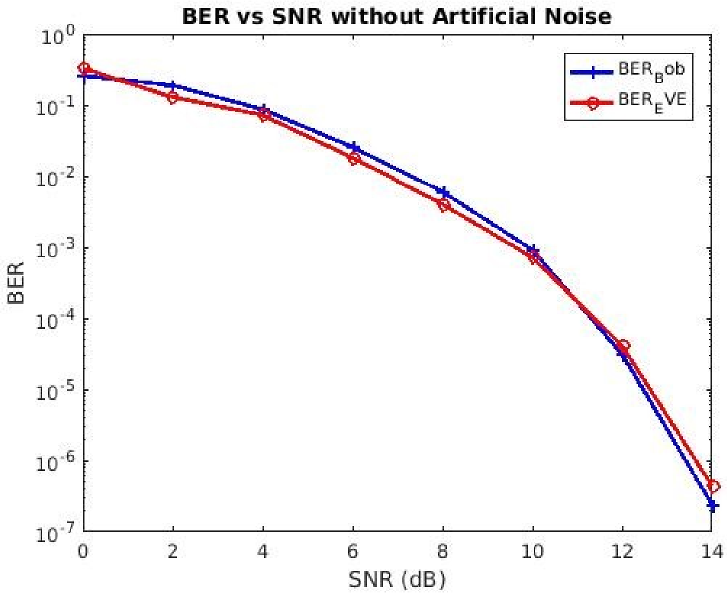


Figure 8.1: Experimental BER curves for Bob (blue) and Eve (red) with no Artificial Noise added.

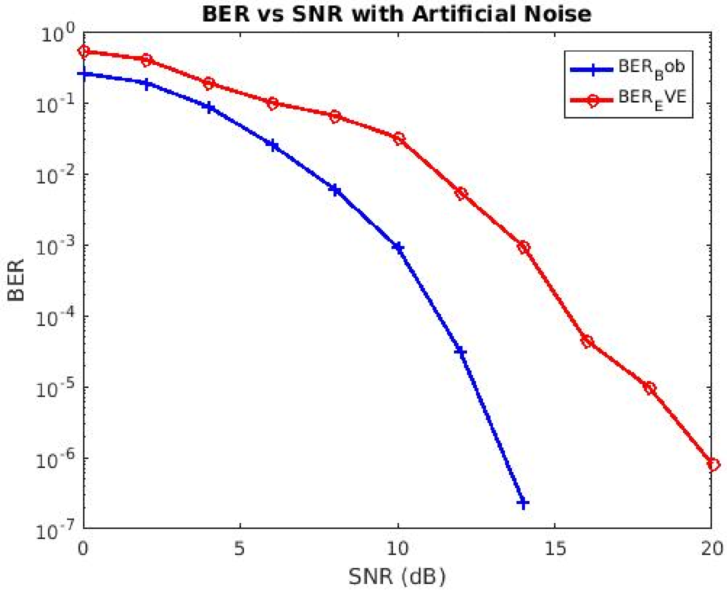


Figure 8.2: Experimental BER curves for Bob (blue) and Eve (red) with Artificial Noise added with a power ratio of $\alpha = 0.5$.

Figure 8.1 confirms that other factors in the testing environment, such as positions of Bob or Eve, do not have a significant impact on their relative performance. Figure 8.2 shows the performance gap between Bob and Eve that is introduced by adding AN.

The BER measurements in Figure 8.2 must have an error associated with them since the BER values for Eve are larger than 0.5 for low values of SNR. These results were collected as part of an intern program assignment and, unfortunately, the source data used for them cannot be recovered. Due to time considerations and the state of the laboratory environment used to conduct these experiments, the results were not able to be modified for this thesis.

Chapter 9

Conclusion

Many PLS techniques are well-researched and just on the edge of adoption into wireless standards. By designing and incorporating these techniques in OTA systems, the practical challenges of their implementation can be addressed. This thesis has demonstrated the design process of two PLS systems that can both operate OTA. Additionally, a methodology for conducting a characterization of one of these systems was described and some initial experimental results were provided.

An area of future work for the AN and PEAC systems is to perform a more strict OTA characterization for a standardized channel model. This will require setting up an OTA experiment in an environment that conforms to a particular channel model by using a channel emulator and computing BER curves to compare against expected results from simulations or theory. A good candidate for this channel model would be an indoor WINNER 3GPP model.

Another area for future work is to increase the array sizes of Alice, Bob, and Eve to compare the experimentally derived relationships between them with the theoretical results of a masked beamformer found in [12]. Some work not included in this thesis was done on examining the situations described in [12] where an eavesdropper with enough antennas can overcome AN. Demonstrating this property OTA would provide more insight into how AN could be practically applied to existing communication systems.

A final recommendation for future work on this topic is in the application of AN to WIFI systems. The 802.11n and 802.11ac standards make use of CSI feedback natively to perform closed-loop MIMO techniques such as transmit beamforming. Therefore, the addition of AN to these systems would be minimally invasive. In fact, since the noise is additive, no modification to the router may be necessary. Instead, a "black box" solution could be developed that is placed between the output SMA of the router and the antenna.

Chapter 10

Appendix A: Matlab Code

10.1 AN Simulation

```matlab
% Artificial Noise Generation with Transmit Beamforming
% Created: 01/2017
% Last Modified: 09/2017
% Decription: This is a simulation of the Masked Beamforming scheme
% described in Secure Transmission with Multiple Antennas I: The MISOME Wiretap Channel by A. Khisti.
% Sources: [1] Secure Transmission with Multiple Antennas I: The MISOME Wiretap Channel by A. Khisti [2] http://www.dsplog.com/2009/04/13/transmit–beamforming/

% Refresh Matlab
close all;
clear;
clc;

```

```
% Parameters
N = 10^4; % Number of Information Symbols
n = 2; % Modulation Order
L = 2^n; % Number of Modulation Points
EbNo_dB = -25:1:30;
EbNo_lin = 10.^(EbNo_dB/10);
EsNo_dB = EbNo_dB+(n-1)*3;
r = 0.9; % Ratio of AN Power to Total Power

% Initializations
errors_bob = zeros(1,length(EbNo_dB));
errors_eve = zeros(1,length(EbNo_dB));

for ii = 1:1:length(EbNo_dB)
% CHANNELS (Alice-Bob, Alice-Eve)
% Rayleigh Fading Channel with Normalized Gain
h_ab = 1/sqrt(2)*(randn(1,N) + 1j*randn(1,N));
h_ae = 1/sqrt(2)*(randn(1,N) + 1j*randn(1,N));
% We assume the channel stays constant over 2 symbol durations
h_ab = repelem(reshape(h_ab,2,N/2),1,2);
h_ae = repelem(reshape(h_ae,2,N/2),1,2);

% TRANSMITTER (Alice)
% QPSK Modulation (using Grey coding)
x = 2*round(rand(1,N))+round(rand(1,N));
b = reshape(dec2bin(x).',1,2*N);
s = zeros(1,N);
for p=1:N
```

```
if x(p) == 0
s(p) = -1;
elseif x(p) == 1
s(p) = -1j;
elseif x(p) == 2
s(p) = 1j;
else
s(p) = 1;
end
end

% Eigenmode Beamformer (TX Beamforming)
s = 1/sqrt(2)*repelem(s,2,1);
bf = [exp(-1j*angle(h_ab(1,:)));exp(-1j*angle(h_ab(2,:)))];
s(1,:) = s(1,:).*bf(1,:);
s(2,:) = s(2,:).*bf(2,:);

% Generate Artificial Noise Based on Bob's Channel
for i = 1:N
w(:,i) = null(h_ab(:,i).');
end
s = sqrt(1-r)*s + sqrt(r)*w;

% Additive White Gaussian Noise
awgn_ab = 10^(-(EsNo_dB(ii))/20)/sqrt(2)*(randn(1,N) + 1j*randn(1,N));
awgn_ae = 10^(-EsNo_dB(ii)/20)/sqrt(2)*(randn(1,N) + 1j*randn(1,N));

% Transmission Experiences Channel Effects
```

```
r_bob = sum(h_ab.*s,1) + awgn_ab;
r_eve = sum(h_ae.*s,1) + awgn_ae;

% INTENDED RECEIVER (Bob)
sHat_bob = r_bob;
% Symbol decisions are made by measuring the phase of the received symbol and
% comparing it to established decision boundaries.
theta_bob = 180/pi*angle(sHat_bob);
xHat_bob = zeros(1,N);
for d=1:N
if -45<=theta_bob(d) && theta_bob(d)<45
xHat_bob(d) = 3;
elseif 45<=theta_bob(d) && theta_bob(d)<135
xHat_bob(d) = 2;
elseif 135<=theta_bob(d) || -135>=theta_bob(d)
xHat_bob(d) = 0;
else
xHat_bob(d) = 1;
end
end
bHat_bob = reshape(dec2bin(xHat_bob).',1,2*N);

% UNINTENDED TRANSMITTER (Eve)
% (1) Equalization with both Bob and Eve's CSI
delta = 1./(h_ae(1,:).*conj(h_ab(1,:))./(abs(h_ab(1,:)))+h_ae(2,:).*conj(h_ab(2,:))./(abs(h_ab(2,:))));
sHat_eve = r_eve.*delta;
```

```

% Symbol decisions are made by measuring the phase of the received symbol and
% comparing it to established decision boundaries.
theta_eve = 180/pi*angle(sHat_eve);
xHat_eve = zeros(1,N);
for d=1:N
if -45<=theta_eve(d) && theta_eve(d)<45
xHat_eve(d) = 3;
elseif 45<=theta_eve(d) && theta_eve(d)<135
xHat_eve(d) = 2;
elseif 135<=theta_eve(d) || -135>=theta_eve(d)
xHat_eve(d) = 0;
else
xHat_eve(d) = 1;
end
end
bHat_eve = reshape(dec2bin(xHat_eve).',1,2*N);

% Count the Errors
errors_bob(ii) = size(find(bHat_bob-b),2);
errors_eve(ii) = size(find(bHat_eve-b),2);

clearvars hEq_eve
end

% Simulation Results
BERsim_bob = errors_bob/(2*N);
```

```
BERsim_eve = errors_eve/(2*N);

% Theoredical Results
p = 1/2 - 1/2*(1+1./EbNo_lin).^(-1/2);
theoryBer_nRx2 = p.^2.*(1+2*(1-p));
theoryBer_rayleigh = 0.5.*(1-sqrt(EbNo_lin./(EbNo_lin+1)));

% Graph Theoretical and Simulated Curves
close all
figure
semilogy(EbNo_dB,theoryBer_nRx2,'p-','LineWidth',2);
hold on
semilogy(EbNo_dB,theoryBer_rayleigh,'x-','LineWidth',2);
semilogy(EbNo_dB,BERsim_bob,'o-','LineWidth',2);
semilogy(EbNo_dB,BERsim_eve,'s-','LineWidth',2);
axis([-25 30 10^-5 1])
grid on
legend('theory SIMO (nTx=1, nRx=2, MRC)','theory SISO (nTx=1, nRx=1, Rayleigh)','sim Bob (nTx=2, nRx=1, TX BF with AN)', 'sim Eve (nTx=2, nRx=1, TX BF with AN)');
xlabel('Eb/No, dB');
ylabel('Bit Error Rate');
% Commented out to avoid interfering with Latex
%title(sprintf('$$\\textbf{BER for QPSK with Artificial Noise }\\mathbf{(\\alpha=%.2f)}\\textbf{ and Transmit Beamforming in a Rayleigh Channel }$$', r), 'interpreter', 'latex');
```

10.2 PEAC Simulation

```
% Dynamic Phase Alamouti Code Simulation
% Created: 10/20/2016
% Last Modified: 09/01/2017
% Decription: This is a simulation of the Dynamic Phase Alamouti scheme
% described in Secure Space—Time Block Coding without Transmitter CSI by T.
    Allen.
% Sources: [1] Secure Space—Time Block Coding without Transmitter CSI by T.
    Allen [2] http://www.dsplog.com/2008/10/16/alamouti—stbc/

% Refresh Matlab
close all;
clear;
clc;

% Parameters
N = 10^6; % Number of Information Symbols
n = 2; % Modulation Order
L = 2^n; % Number of Modulation Points
EbNo_dB = —25:1:30;
EbNo_lin = 10.^(EbNo_dB/10);
EsNo_dB = EbNo_dB+(n—1)∗3;

% Initializations
errors_bob = zeros(1,length(EsNo_dB));
errors_eve1 = zeros(1,length(EsNo_dB));
errors_eve2 = zeros(1,length(EsNo_dB));
```

```

for ii = 1:1:length(EsNo_dB)
% TRANSMITTER (Alice)
% create QPSK symbols (using Grey coding)
x = 2*round(rand(1,N))+round(rand(1,N));
b = reshape(dec2bin(x).',1,2*N);
x_m = zeros(1,N);
for p=1:N
if x(p) == 0
x_m(p) = -1;
elseif x(p) == 1
x_m(p) = -1j;
elseif x(p) == 2
x_m(p) = 1j;
else
x_m(p) = 1;
end
end

% Alamouti STBC
s = 1/sqrt(2)*repelem(reshape(x_m,2,N/2),1,2);
temp = s;
s(1,2:2:end) = -conj(temp(2,2:2:end)); % S1 = [X1, -X2*]
s(2,2:2:end) = conj(temp(1,2:2:end));  % S2 = [X2,  X1*]

% Phase Shifter Fed by a psuedo-random sequence
c = round((L-1)*rand(1,N));
c = repelem(reshape(c,2,N/2),1,2);
```

```
% By zeroing out the first antenna's phase shifts, the diversity order
% of Eve's BER curve should be increased by 1.
theta_shift = c*2*pi/L; % [theta1 theta1 ...; theta2 theta2 ...]
s = s.*exp(1j*theta_shift);

% CHANNELS (Alice-Bob, Alice-Eve)
% Rayleigh Fading Channel with Normalized Gain
h_ab = 1/sqrt(2)*(randn(1,N) + 1j*randn(1,N));
h_ae = 1/sqrt(2)*(randn(1,N) + 1j*randn(1,N));
% We assume the channel stays constant over 2 symbol durations
h_ab = repelem(reshape(h_ab,2,N/2),1,2);
h_ae = repelem(reshape(h_ae,2,N/2),1,2);

% Additive White Gaussian Noise
awgn_ab = 10^(-EsNo_dB(ii)/20)/sqrt(2)*(randn(1,N) + 1j*randn(1,N));
awgn_ae = 10^(-EsNo_dB(ii)/20)/sqrt(2)*(randn(1,N) + 1j*randn(1,N));

% Transmission Experiences Channel Effects
r_bob = sum(h_ab.*s,1) + awgn_ab;
r_eve = sum(h_ae.*s,1) + awgn_ae;

% INTENDED RECEIVER (Bob)
% (1) Convert received signal to format required to isolate the transmitted symbols
r_bob = repelem(reshape(r_bob,2,N/2),1,2); % [r1 r1 ... ; r2 r2 ...]
r_bob(2,:) = conj(r_bob(2,:)); % [r1 r1 ... ; r2* r2*...]

% (2) Create the equalization matrix, undoing the phase shift with
% knowledge of the psuedo-random sequence
theta_deshift = zeros(2,N);
theta_deshift(1,1:2:end) = -c(1,1:2:end)*2*pi/L;
theta_deshift(1,2:2:end) = -c(2,2:2:end)*2*pi/L;
theta_deshift(2,1:2:end) = c(2,1:2:end)*2*pi/L;
theta_deshift(2,2:2:end) = c(1,2:2:end)*2*pi/L; % [-theta1 -theta2 ...; theta2 theta1 ...]
hEq_bob = zeros(2,N);
hEq_bob(:,1:2:end) = h_ab(:,1:2:end); % [h1 0 ...; h2 0 ...]
hEq_bob(:,2:2:end) = kron(ones(1,N/2),[1;-1]).*flipud(h_ab(:,1:2:end)); % [h1 h2 ...; h2 -h1 ...]
hEq_bob(1,:) = conj(hEq_bob(1,:)); %  [h1* h2* ...; h2 -h1 ...]
hEq_bob = hEq_bob.*exp(1j*theta_deshift); % [h1*exp(-jtheta1) h2*exp(-jtheta2) ...; h2exp(jtheta2) h1exp(jtheta1) ...]
hEqPower_bob = sum(hEq_bob.*conj(hEq_bob),1); % hEqPower = [abs(h1)^2 + abs(h2)^2, 0 ...; 0, abs(h1)^2 + abs(h2)^2 ...] = hermitian(H)*H

% (3) Determine the estimate of the received signal
sHat_bob = sum(hEq_bob.*r_bob,1)./hEqPower_bob;

% (4) QPSK Demodulation
theta_bob = 180/pi*angle(sHat_bob);
xHat_bob = zeros(1,N);
for d=1:N
if -45<=theta_bob(d) && theta_bob(d)<45
xHat_bob(d) = 3;
elseif 45<=theta_bob(d) && theta_bob(d)<135
xHat_bob(d) = 2;
```

```matlab
elseif 135<=theta_bob(d) || -135>=theta_bob(d)
xHat_bob(d) = 0;
else
xHat_bob(d) = 1;
end
end

bHat_bob = reshape(dec2bin(xHat_bob).',1,2*N);

% Maximum Likelihood Receiver Detailed in [1]

% UNINTENDED Receiver with Matched Filterbank (Eve2)
% (1) Convert received signal to format required to isolate the transmitted symbols
r_eve2 = repelem(reshape(r_eve,2,N/2),1,2); % [r1 r1 ... ; r2 r2 ...]
r_eve2(2,:) = conj(r_eve2(2,:)); % [r1 r1 ... ; r2* r2*...]

% (2) Eve knows the scheme being used, but has no knowledge of the
% psuedo-random sequence.  Therefore, she tries to estimate the symbols
% by employing L^2 parallel processing branches (matched filters),
% comparing the result to estimated symbols, and minimizing the
% mean-squared error.
theta_guess = 0:2*pi/L:(2*pi*(L-1))/L; % All possible shift values
[th1, th2] = meshgrid(theta_guess);
%th2 = zeros(4,4);
theta_guess = reshape(cat(2,th1,th2),[],2).'; % All possible combinations of shift values
theta_guess = reshape(theta_guess,2,1,[]);
theta_guess = repelem(theta_guess,1,N,1); % [theta1 ...; theta2 ...] Across L^2 Entries in 3D
theta_eve2 = zeros(2,N,L^2);
theta_eve2(1,1:2:end,:) = -theta_guess(1,1:2:end,:);
theta_eve2(1,2:2:end,:) = -theta_guess(2,2:2:end,:);
theta_eve2(2,1:2:end,:) = theta_guess(2,1:2:end,:);
theta_eve2(2,2:2:end,:) = theta_guess(1,2:2:end,:); % [-theta1 -theta2 ...; theta2 theta1 ...] Across L^2 Entries in 3D
hEq_eve2 = zeros(2,N);
hEq_eve2(:,1:2:end) = h_ae(:,1:2:end); % [h1 0 ...; h2 0 ...]
hEq_eve2(:,2:2:end) = kron(ones(1,N/2),[1;-1]).*flipud(h_ae(:,1:2:end)); % [h1 h2 ...; h2 -h1 ...]
hEq_eve2(1,:) = conj(hEq_eve2(1,:)); %  [h1* h2* ...; h2 -h1 ...]
hEq_eve2 = repmat(hEq_eve2,1,1,L^2); %  [h1* h2* ...; h2 -h1 ...] Across L^2 Entries in 3D
hEq_eve2 = hEq_eve2.*exp(1j*theta_eve2); % [h1*exp(-jtheta1) h2*exp(-jtheta2) ...; h2exp(jtheta2) h1exp(jtheta1) ...] Across L^2 Entries in 3D
hEqPower_eve2 = sum(hEq_eve2.*conj(hEq_eve2),1); % hEqPower = [abs(h1)^2 + abs(h2)^2, 0 ...; 0, abs(h1)^2 + abs(h2)^2 ...] = hermitian(H)*H

% Compute Estimates of the transmitted symbols for each theta pair
sTilda_eve2 = squeeze(sum(hEq_eve2.*repmat(r_eve2,1,1,L^2),1)./hEqPower_eve2);

% Compare Estimates with estimated symbols and choose the one with the
% smallest error
```

```
sHat_eve2 = sign(real(sTilda_eve2))+sign(imag(sTilda_eve2)); % Estimated Symbols
dist = abs(sHat_eve2 - sTilda_eve2);
[dummy,index] = min(dist,[],2);
for kk = 1:length(sHat_eve2)
sFinal_eve2(kk) = sHat_eve2(kk,index(kk));
end

% (3) QPSK Demodulation
theta_eve2 = 180/pi*angle(sFinal_eve2);
xHat_eve2 = zeros(1,N);
for d=1:N
if -45<=theta_eve2(d) && theta_eve2(d)<45
xHat_eve2(d) = 3;
elseif 45<=theta_eve2(d) && theta_eve2(d)<135
xHat_eve2(d) = 2;
elseif 135<=theta_eve2(d) || -135>=theta_eve2(d)
xHat_eve2(d) = 0;
else
xHat_eve2(d) = 1;
end
end

bHat_eve2 = reshape(dec2bin(xHat_eve2).',1,2*N);

% Count the Errors
errors_bob(ii) = size(find(bHat_bob-b),2);
errors_eve2(ii) = size(find(bHat_eve2-b),2);
```

```
clearvars hEq_eve
end

% Simulation Results
BERsim_bob = errors_bob/(2*N);
BERsim_eve2 = errors_eve2/(2*N);

% Theoredical Results
% Rayleigh Channel [1]
theoryBer_nRx1 = 0.5.*(1-1*(1+1./EbNo_lin).^(-0.5));
% MRC Algorithm [1]
p = 1/2 - 1/2*(1+1./(2*EbNo_lin)).^(-1/2);
theoryBerMRC_nRx2 = p.^2.*(1+2*(1-p));
% Alamouti Coding [1]
pAlamouti = 1/2 - 1/2*(1+2./EbNo_lin).^(-1/2);
theoryBerAlamouti_nTx2_nRx1 = pAlamouti.^2.*(1+2*(1-pAlamouti));

close all
figure
semilogy(EsNo_dB,theoryBer_nRx1,'p-','LineWidth',2);
hold on
semilogy(EsNo_dB,theoryBerMRC_nRx2,'d-','LineWidth',2);
semilogy(EsNo_dB,theoryBerAlamouti_nTx2_nRx1,'+-','LineWidth',2);
semilogy(EsNo_dB,BERsim_bob,'o-','LineWidth',2);
semilogy(EsNo_dB,BERsim_eve2,'s-','LineWidth',2);
axis([-25 30 10^-5 1])
grid on
```

```
legend('theory (nTx=1,nRx=1)', 'theory (nTx=1,nRx=2, MRC)', 'theory (nTx=2, nRx=1, Alamouti)', 'sim Bob (nTx=2, nRx=1, Alamouti)', 'sim Eve (nTx=2, nRx=1, Matched Filterbank)');
xlabel('Eb/No, dB');
ylabel('Bit Error Rate');
title('BER with QPSK modulation and Phase Enciphered Alamouti Coding ( Rayleigh channel)')
```

Bibliography

[1] A. D. Wyner, "The wire-tap channel," *The Bell System Technical Journal*, vol. 54, no. 8, pp. 1355–1387, Oct 1975.

[2] D. J. C. MacKay, *Information theory, inference, and learning algorithms*. Cambridge University Press, 2005, vol. 7.2.

[3] E. Biglieri, *MIMO wireless communications*. Cambridge University Press, 2010.

[4] Y. S. Shiu, S. Y. Chang, H. C. Wu, S. C. H. Huang, and H. H. Chen, "Physical layer security in wireless networks: a tutorial," *IEEE Wireless Communications*, vol. 18, no. 2, pp. 66–74, April 2011.

[5] X. Chen, D. W. K. Ng, W. Gerstacker, and H. H. Chen, "A survey on multiple-antenna techniques for physical layer security," *IEEE Communications Surveys Tutorials*, vol. PP, no. 99, pp. 1–1, 2016.

[6] v3l0c1r4pt0r, "Rtl-sdr fm radio receiver with gnu radio companion," May 2016. [Online]. Available: http://www.instructables.com/id/RTL-SDR-FM-radio-receiver-with-GNU-Radio-Companion/

[7] T. Rondeau, "Blog - control loop gain values," Aug 2011. [Online]. Available: http://gnuradio.squarespace.com/blog/2011/8/13/control-loop-gain-values.html;jsessionid=51490B8FC8D317DF71A9FAF2B1A4688B.v5-web005

[8] "Guided tutorials," Mar 2017. [Online]. Available: https://wiki.gnuradio.org/index.php/Guided_Tutorials

[9] N. Pandeya, "About usrp bandwidths and sampling rates," May 2016. [Online]. Available: https://kb.ettus.com/About_USRP_Bandwidths_and_Sampling_Rates

[10] A. Mukherjee, "Physical-layer security in the internet of things: Sensing and communication confidentiality under resource constraints," *Proceedings of the IEEE*, vol. 103, no. 10, pp. 1747–1761, 2015.

[11] A. Mukherjee, S. A. A. Fakoorian, J. Huang, and A. L. Swindlehurst, "Principles of physical layer security in multiuser wireless networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1550–1573, Third 2014.

[12] A. Khisti and G. W. Wornell, "Secure transmission with multiple antennas part ii: The mimome wiretap channel," *IEEE Transactions on Information Theory*, vol. 56, no. 11, pp. 5515–5532, Nov 2010.

[13] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov, "The first collision for full sha-1." *IACR Cryptology ePrint Archive*, vol. 2017, p. 190, 2017.

[14] R. Cox, "Lessons from the debian/openssl fiasco posted on wednesday, may 21, 2008." May 2008. [Online]. Available: https://research.swtch.com/openssl

[15] "Usn-612-1: Openssl vulnerability," May 2008. [Online]. Available: https://www.ubuntu.com/usn/usn-612-1/

[16] A. J. Paulraj, D. A. Gore, R. U. Nabar, and H. Bolcskei, "An overview of mimo communications-a key to gigabit wireless," *Proceedings of the IEEE*, vol. 92, no. 2, pp. 198–218, 2004.

[17] J. Mitola, "Software radios: Survey, critical evaluation and future directions," *IEEE Aerospace and Electronic Systems Magazine*, vol. 8, no. 4, pp. 25–36, April 1993.

[18] T. Ulversoy, "Software defined radio: Challenges and opportunities," *IEEE Communications Surveys Tutorials*, vol. 12, no. 4, pp. 531–550, Fourth 2010.

[19] M. P. Daly and J. T. Bernhard, "Beamsteering in pattern reconfigurable arrays using directional modulation," *IEEE Transactions on Antennas and Propagation*, vol. 58, no. 7, pp. 2259–2265, 2010.

[20] V. Pellegrini, F. Principe, G. De Mauro, R. Guidi, V. Martorelli, and R. Cioni, "Cryptographically secure radios based on directional modulation," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2014, pp. 8163–8167.

[21] B. R. Jordan, E. R. Tollefson, and J. D. Gaeddert, "Characterization of out-phased array linearized signaling (opals)," in *IEEE International Symposium on Phased Array Systems and Technology*. IEEE, 2016, pp. 1–7.

[22] E. Tollefson, B. R. Jordan, and J. D. Gaeddert, "Out-phased array linearized signaling (opals): A practical approach to physical layer encryption," in *Military Communications Conference, MILCOM 2015-2015 IEEE*. IEEE, 2015, pp. 294–299.

[23] C. L. K. Ngassa, R. Molière, F. Delaveau, A. Sibille, and N. Shapira, "Secret key generation scheme from wifi and lte reference signals," *Analog Integrated Circuits and Signal Processing*, vol. 91, no. 2, pp. 277–292, 2017.

[24] A. Khisti and G. W. Wornell, "Secure transmission with multiple antennas i: The misome wiretap channel," *IEEE Transactions on Information Theory*, vol. 56, no. 7, pp. 3088–3104, July 2010.

[25] Y. Liu, H. H. Chen, and L. Wang, "Physical layer security for next generation wireless networks: Theories, technologies, and challenges," *IEEE Communications Surveys Tutorials*, vol. 19, no. 1, pp. 347–376, Firstquarter 2017.

[26] S. A. A. Fakoorian and A. L. Swindlehurst, “Optimal power allocation for gsvd-based beamforming in the mimo gaussian wiretap channel,” in *2012 IEEE International Symposium on Information Theory Proceedings*, July 2012, pp. 2321–2325.

[27] H. Reboredo, J. Xavier, and M. R. D. Rodrigues, “Filter design with secrecy constraints: The mimo gaussian wiretap channel,” *IEEE Transactions on Signal Processing*, vol. 61, no. 15, pp. 3799–3814, Aug 2013.

[28] R. Negi and S. Goel, “Secret communication using artificial noise,” in *VTC-2005-Fall. 2005 IEEE 62nd Vehicular Technology Conference, 2005.*, vol. 3, Sept 2005, pp. 1906–1910.

[29] F. Kaltenberger, H. Jiang, M. Guillaud, and R. Knopp, “Relative channel reciprocity calibration in mimo/tdd systems,” in *2010 Future Network Mobile Summit*, June 2010, pp. 1–10.

[30] S. M. Bellovin, “Frank miller: Inventor of the one-time pad,” *Cryptologia*, vol. 35, no. 3, pp. 203–222, 2011.

[31] C. E. Shannon, “Communication theory of secrecy systems,” *The Bell System Technical Journal*, vol. 28, no. 4, pp. 656–715, Oct 1949.

[32] S. Leung-Yan-Cheong and M. Hellman, “The gaussian wire-tap channel,” *IEEE Transactions on Information Theory*, vol. 24, no. 4, pp. 451–456, Jul 1978.

[33] K. R. Liu, *Cooperative communications and networking*. Cambridge university press, 2009.

[34] S. M. Alamouti, “A simple transmit diversity technique for wireless communications,” *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 8, pp. 1451–1458, Oct 1998.

[35] T. Allen, J. Cheng, and N. Al-Dhahir, “Secure space-time block coding without transmitter csi,” *IEEE Wireless Communications Letters*, vol. 3, no. 6, pp. 573–576, Dec 2014.

[36] “Outoftreemodules,” May 2017. [Online]. Available: https://wiki.gnuradio.org/index.php/OutOfTreeModules

[37] E. ETSI, “300 908 (gsm 05.02), digital cellular telecommunications system,” *Multiplexing and Multiple Access on the Radio Path*.

[38] F. Harris, *Multirate signal processing for communication systems*. Prentice Hall PTR, 2004.

[39] “X300/x310,” Mar 2017. [Online]. Available: https://kb.ettus.com/X300/X310

[40] M. R. Cribbs, “Multiple-input multiple-output wavelet packet modulation based software-defined radio transceiver design,” Ph.D. dissertation, Monterey, California: Naval Postgraduate School, 2015.

[41] R. A. Shafik, M. S. Rahman, and A. R. Islam, “On the extended relationships among evm, ber and snr as performance metrics,” in *Electrical and Computer Engineering, 2006. ICECE’06. International Conference on*. IEEE, 2006, pp. 408–411.